

**ROMA Connect**

# **User Guide**

**Issue**            05  
**Date**             2024-05-07



**Copyright © Huawei Cloud Computing Technologies Co., Ltd. 2024. All rights reserved.**

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Cloud Computing Technologies Co., Ltd.

## **Trademarks and Permissions**



HUAWEI and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

## **Notice**

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

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

---

# Contents

---

|  |           |
|--|-----------|
| <b>1 Usage Process.....</b>                                    | <b>1</b>  |
| <b>2 Instances.....</b>  | <b>3</b>  |
| 2.1 Creating a ROMA Connect Instance.....                      | 3         |
| 2.1.1 Preparing Required Resources.....                        | 3         |
| 2.1.2 Creating an Instance.....                                | 5         |
| 2.2 Managing Instances.....                                    | 9         |
| 2.2.1 Viewing Details of an Instance.....                      | 9         |
| 2.2.2 Modifying Instance Configuration Parameters.....         | 11        |
| 2.2.3 Creating a Data Dictionary.....                          | 16        |
| 2.2.4 Importing and Exporting Assets.....                      | 17        |
| 2.3 Restoring/Migrating Integration Assets.....                | 19        |
| 2.3.1 Overview.....  | 19        |
| 2.3.2 Preparations.....  | 22        |
| 2.3.3 Modifying Instance Configurations.....                   | 23        |
| 2.3.4 Importing Integration Assets.....                        | 23        |
| 2.3.5 Modifying Integration Application Configurations.....    | 24        |
| 2.3.6 Modifying Data Source Configurations.....                | 24        |
| 2.3.7 Modifying API Configurations.....                        | 25        |
| 2.3.8 Modifying Device Configurations.....                     | 28        |
| 2.3.9 Service Interconnection Adaptation and Verification..... | 29        |
| <b>3 Integration Application Management.....</b>               | <b>31</b> |
| 3.1 Creating an Integration Application.....                   | 31        |
| 3.2 Configuring Integration Application Authorization.....     | 33        |
| <b>4 Data Source Management.....</b>                           | <b>36</b> |
| 4.1 Data Sources Supported by ROMA Connect.....                | 37        |
| 4.2 Connecting to an API Data Source.....                      | 39        |
| 4.3 Connecting to an ActiveMQ Data Source.....                 | 43        |
| 4.4 Connecting to an ArtemisMQ Data Source.....                | 45        |
| 4.5 Connecting to a DB2 Data Source.....                       | 47        |
| 4.6 Connecting to a DIS Data Source.....                       | 50        |
| 4.7 Connecting to a DWS Data Source.....                       | 52        |
| 4.8 Connecting to the DM Data Source.....                      | 53        |

|  |            |
|--|------------|
| 4.9 Connecting to a Gauss100 Data Source.....                    | 55         |
| 4.10 Connecting to an FTP Data Source.....                       | 57         |
| 4.11 Connecting to an HL7 Data Source.....                       | 59         |
| 4.12 Connecting to a HANA Data Source.....                       | 61         |
| 4.13 Connecting to a HIVE Data Source.....                       | 63         |
| 4.14 Connecting to an LDAP Data Source.....                      | 64         |
| 4.15 Connecting to an IBM MQ Data Source.....                    | 66         |
| 4.16 Connecting to a Kafka Data Source.....                      | 69         |
| 4.17 Connecting to a MySQL Data Source.....                      | 70         |
| 4.18 Connecting to a MongoDB Data Source.....                    | 74         |
| 4.19 Connecting to an MQS Data Source.....                       | 76         |
| 4.20 Connecting to an MRS Hive Data Source.....                  | 78         |
| 4.21 Connecting to an MRS HDFS Data Source.....                  | 80         |
| 4.22 Connecting to an MRS HBase Data Source.....                 | 82         |
| 4.23 Connecting to an MRS Kafka Data Source.....                 | 84         |
| 4.24 Connecting to an OBS Data Source.....                       | 85         |
| 4.25 Connecting to an Oracle Data Source.....                    | 87         |
| 4.26 Connecting to a PostgreSQL Data Source.....                 | 90         |
| 4.27 Connecting to a Redis Data Source.....                      | 92         |
| 4.28 Connecting to a RabbitMQ Data Source.....                   | 94         |
| 4.29 Connecting to a RocketMQ Data Source.....                   | 95         |
| 4.30 Connecting to an SAP Data Source.....                       | 97         |
| 4.31 Connecting to an SNMP Data Source.....                      | 100        |
| 4.32 Connecting to a SQL Server Data Source.....                 | 102        |
| 4.33 Connecting to a GaussDB(for MySQL) Data Source.....         | 104        |
| 4.34 Connecting to a WebSocket Data Source.....                  | 107        |
| 4.35 Connecting to a Custom Data Source.....                     | 109        |
| <b>5 Data Integration Guide.....</b>                             | <b>111</b> |
| 5.1 Usage Introduction.....                                      | 111        |
| 5.2 Connecting to Data Sources.....                              | 113        |
| 5.3 Creating a Common Data Integration Task.....                 | 116        |
| 5.3.1 Configuring Basic Information.....                         | 116        |
| 5.3.2 Configuring Source Information.....                        | 120        |
| 5.3.3 Configuring Destination Information.....                   | 214        |
| 5.3.4 Configuring a Data Mapping Rule.....                       | 258        |
| 5.3.5 (Optional) Configuring Fault Information Storage.....      | 260        |
| 5.3.6 (Optional) Configuring the Post-Integration Operation..... | 261        |
| 5.4 Creating a Composite Data Integration Task.....              | 263        |
| 5.4.1 Configuring Oracle CDC (LogMiner).....                     | 263        |
| 5.4.2 Configuring Oracle CDC (XStream).....                      | 268        |
| 5.4.3 Configuring the MySQL CDC (Binlog).....                    | 277        |
| 5.4.4 Configuring SQL Server CDC.....                            | 279        |

|  |            |
|--|------------|
| 5.4.5 Creating a Composite Task.....                       | 283        |
| 5.5 Creating a Flow Data Integration Task.....             | 289        |
| 5.5.1 Configuring a Flow Task Process.....                 | 289        |
| 5.5.2 Configuring Source Information.....                  | 294        |
| 5.5.3 Configuring Destination Information.....             | 334        |
| 5.5.4 Configuring Mapping Rules.....                       | 354        |
| 5.6 Starting or Stopping a Data Integration Task.....      | 357        |
| 5.7 Managing a Data Integration Task.....                  | 358        |
| 5.7.1 Viewing Data Integration Tasks.....                  | 358        |
| 5.7.2 Importing or Exporting Data Integration Tasks.....   | 360        |
| 5.7.3 Appendix: Quartz Cron Expression Configuration.....  | 360        |
| 5.8 Connectors.....  | 363        |
| 5.8.1 Creating a Connector.....                            | 363        |
| 5.8.2 Publishing Connectors.....                           | 366        |
| <b>6 Service Integration Guide.....</b>                    | <b>368</b> |
| 6.1 Usage Introduction.....                                | 368        |
| 6.2 Exposing an API.....                                   | 371        |
| 6.2.1 Creating an API Group.....                           | 371        |
| 6.2.2 (Optional) Creating a Load Balance Channel.....      | 372        |
| 6.2.3 Creating an API.....                                 | 375        |
| 6.2.4 Debugging an API.....                                | 389        |
| 6.2.5 (Optional) Creating an Environment and Variable..... | 390        |
| 6.2.6 Publishing an API.....                               | 392        |
| 6.2.7 Binding Domain Names.....                            | 393        |
| 6.2.8 (Optional) Authorizing Credentials to Call APIs..... | 394        |
| 6.3 Exposing a Function API.....                           | 395        |
| 6.3.1 Creating a Function Backend.....                     | 395        |
| 6.3.2 Publishing a Function API.....                       | 397        |
| 6.3.3 Binding Domain Names.....                            | 399        |
| 6.3.4 (Optional) Authorizing Credentials to Call APIs..... | 401        |
| 6.4 Exposing a Data API.....                               | 402        |
| 6.4.1 Connecting to a Data Source.....                     | 402        |
| 6.4.2 Creating a Data Backend.....                         | 403        |
| 6.4.3 Publishing a Data API.....                           | 407        |
| 6.4.4 Binding Domain Names.....                            | 409        |
| 6.4.5 (Optional) Authorizing Credentials to Call APIs..... | 410        |
| 6.5 Calling an API.....                                    | 411        |
| 6.5.1 Calling an Open API.....                             | 412        |
| 6.5.2 Configuring CORS for APIs.....                       | 416        |
| 6.5.3 Viewing API Calling Statistics.....                  | 420        |
| 6.5.4 Viewing API Call Logs.....                           | 420        |
| 6.5.5 Appendix: API Error Codes.....                       | 423        |

|  |            |
|--|------------|
| 6.5.6 Response Headers.....  | 426        |
| 6.6 Managing APIs.....   | 428        |
| 6.6.1 Taking an API Offline.....   | 428        |
| 6.6.2 Importing and Exporting APIs.....                                  | 428        |
| 6.6.3 Adding an SSL Certificate.....                                     | 431        |
| 6.6.4 Adding a Credential for Simple Authentication.....                 | 433        |
| 6.6.5 Appendix: Extended Swagger Definition of APIs.....                 | 433        |
| 6.7 Managing Custom Backends.....  | 449        |
| 6.7.1 Taking a Custom Backend Offline.....                               | 449        |
| 6.7.2 Importing and Exporting Custom Backends.....                       | 450        |
| 6.7.3 Adding Public Configurations for Custom Backends.....              | 452        |
| 6.7.4 Appendix: Swagger Extended Definitions for Custom Backends.....    | 453        |
| 6.8 Configuring API Control Policies.....                                | 457        |
| 6.8.1 Configuring API Request Throttling.....                            | 457        |
| 6.8.2 Configuring API Access Control.....                                | 460        |
| 6.8.3 Configuring API Credential Quotas.....                             | 462        |
| 6.8.4 Configuring API Credential Access Control.....                     | 463        |
| 6.9 Configuring API Plug-in Policies.....                                | 464        |
| 6.9.1 Configuring CORS.....  | 464        |
| 6.9.2 Configuring HTTP Response Header Management.....                   | 467        |
| 6.9.3 Configuring Request Throttling 2.0.....                            | 470        |
| 6.9.4 Configuring Kafka Log Push.....                                    | 476        |
| 6.9.5 Configuring Circuit Breakers.....                                  | 481        |
| 6.9.6 Configuring A Third-Party Authorizer.....                          | 489        |
| 6.10 Configuring a Custom Authorizer.....                                | 493        |
| 6.10.1 Creating a Frontend Custom Authorizer.....                        | 493        |
| 6.10.2 Creating a Backend Custom Authorizer.....                         | 497        |
| 6.11 Configuring Signature Verification for Backend Services.....        | 499        |
| 6.12 Configuring API Cascading.....                                      | 501        |
| <b>7 Service Integration Guide (Old Edition).....</b>                    | <b>508</b> |
| 7.1 Usage Introduction.....  | 508        |
| 7.2 Exposing APIs.....   | 510        |
| 7.2.1 Creating an API Group.....   | 511        |
| 7.2.2 (Optional) Creating a Load Balance Channel.....                    | 512        |
| 7.2.3 Creating an API.....   | 515        |
| 7.2.4 Debugging an API.....  | 530        |
| 7.2.5 (Optional) Creating the Environment and Environment Variables..... | 532        |
| 7.2.6 Publishing an API.....   | 533        |
| 7.2.7 Binding Domain Names.....  | 534        |
| 7.2.8 (Optional) Authorizing Applications to Call an API.....            | 535        |
| 7.3 Creating and Exposing Data APIs.....                                 | 537        |
| 7.3.1 Connecting to Data Sources.....                                    | 537        |

|   |            |
|---|------------|
| 7.3.2 Creating a Data API.....  | 538        |
| 7.3.3 Binding Domain Names.....                                       | 544        |
| 7.3.4 (Optional) Authorizing Apps to Call an API.....                 | 546        |
| 7.4 Creating and Exposing Function APIs.....                          | 547        |
| 7.4.1 Creating a Function API.....                                    | 547        |
| 7.4.2 Binding Domain Names.....                                       | 551        |
| 7.4.3 (Optional) Authorizing Apps to Call an API.....                 | 552        |
| 7.5 Calling an API.....   | 553        |
| 7.5.1 Calling an Open API.....  | 553        |
| 7.5.2 Configuring CORS for APIs.....                                  | 558        |
| 7.5.3 Viewing API Calling Statistics.....                             | 562        |
| 7.5.4 Viewing API Call Logs.....                                      | 562        |
| 7.5.5 Appendix: API Error Codes.....                                  | 565        |
| 7.5.6 Response Headers.....   | 568        |
| 7.6 Managing APIs.....  | 570        |
| 7.6.1 Taking an API Offline.....                                      | 570        |
| 7.6.2 Importing and Exporting APIs.....                               | 570        |
| 7.6.3 Configuring an API Scheduled Task.....                          | 573        |
| 7.6.4 SSL Certificate Management.....                                 | 575        |
| 7.6.5 Appendix: Extended Swagger Definition of APIs.....              | 577        |
| 7.7 Managing Custom Backends.....                                     | 593        |
| 7.7.1 Taking a Custom Backend Offline.....                            | 593        |
| 7.7.2 Importing and Exporting Custom Backends.....                    | 593        |
| 7.7.3 Adding Public Configurations for Custom Backends.....           | 595        |
| 7.7.4 Appendix: Swagger Extended Definitions for Custom Backends..... | 597        |
| 7.8 Managing Control Policies.....                                    | 600        |
| 7.8.1 Configuring a Request Throttling Policy.....                    | 601        |
| 7.8.2 Configuring an Access Control Policy.....                       | 604        |
| 7.8.3 Configuring a Client Quota Policy.....                          | 606        |
| 7.8.4 Configuring a Client Access Control Policy.....                 | 607        |
| 7.9 Managing Plug-ins.....  | 608        |
| 7.9.1 Using Plug-ins.....   | 608        |
| 7.9.2 CORS Plug-in.....   | 610        |
| 7.9.3 Kafka Log Push Plug-in.....                                     | 611        |
| 7.9.4 HTTP Response Header Management Plug-in.....                    | 614        |
| 7.9.5 Circuit Breaker Plug-in.....                                    | 616        |
| 7.10 Configuring a Custom Authorizer.....                             | 622        |
| 7.10.1 Creating a Frontend Custom Authorizer.....                     | 622        |
| 7.10.2 Creating a Backend Custom Authorizer.....                      | 626        |
| 7.11 Configuring Signature Verification for Backend Services.....     | 629        |
| 7.12 Configuring API Cascading.....                                   | 631        |
| <b>8 Message Integration Guide.....</b>                               | <b>638</b> |

|  |            |
|--|------------|
| 8.1 Usage Introduction.....                                      | 638        |
| 8.2 Creating a Topic.....  | 639        |
| 8.3 (Optional) Granting Permissions for Topics.....              | 641        |
| 8.4 Connecting to a Topic.....                                   | 642        |
| 8.5 Topic Management.....  | 646        |
| 8.5.1 Viewing Message Body.....                                  | 646        |
| 8.5.2 Importing and Exporting Topics.....                        | 647        |
| 8.6 Migrating Kafka Services.....                                | 649        |
| <b>9 Device Integration Guide.....</b>                           | <b>652</b> |
| 9.1 Usage Introduction.....                                      | 652        |
| 9.2 Creating a Product.....                                      | 654        |
| 9.3 Registering a Device.....                                    | 661        |
| 9.4 Connecting Devices to ROMA Connect.....                      | 663        |
| 9.4.1 Connecting MQTT Devices.....                               | 663        |
| 9.4.2 Connecting Modbus Devices.....                             | 665        |
| 9.4.3 Connecting OPC UA Devices.....                             | 672        |
| 9.5 Product Management.....                                      | 678        |
| 9.5.1 Viewing Product Information.....                           | 678        |
| 9.5.2 Importing and Exporting Products.....                      | 679        |
| 9.5.3 Creating a Product Template.....                           | 682        |
| 9.6 Device Management.....                                       | 685        |
| 9.6.1 Viewing Devices.....                                       | 686        |
| 9.6.2 Importing and Exporting Devices.....                       | 691        |
| 9.6.3 Creating a Device Group.....                               | 693        |
| 9.7 Rule Engine.....   | 695        |
| 9.7.1 Configuring Data Forwarding Rules.....                     | 695        |
| 9.7.2 Importing and Exporting Rules.....                         | 702        |
| 9.8 Subscription Management.....                                 | 704        |
| 9.8.1 Subscribing to Device Notifications.....                   | 704        |
| 9.8.2 Appendix: Packets of Subscribed Notification Messages..... | 705        |
| <b>10 Increasing Resource Quota.....</b>                         | <b>707</b> |
| <b>11 Audit Logs.....</b>  | <b>709</b> |
| <b>12 Monitoring Metrics.....</b>                                | <b>719</b> |
| <b>13 Permissions.....</b>                                       | <b>728</b> |
| 13.1 Assigning ROMA Connect Permissions.....                     | 728        |
| 13.2 ROMA Connect Custom Policies.....                           | 729        |



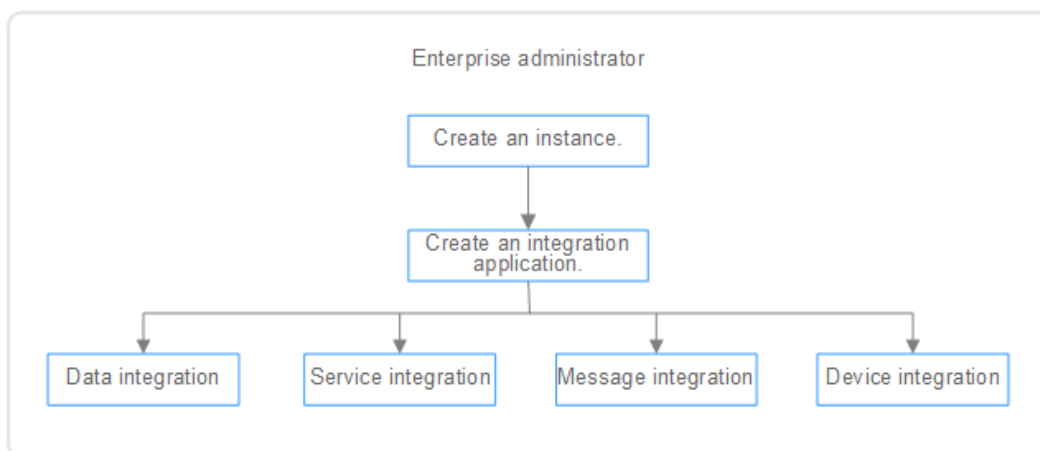
# 1 Usage Process

## Process for Using ROMA Connect

ROMA Connect is a full-stack integration platform that connects applications and data to provide data, API, message, and device integration capabilities. It applies to common enterprise system integration scenarios.

The following figure shows the process for using ROMA Connect.

**Figure 1-1** Process for using ROMA Connect



1. **Create an instance.**  
On the ROMA Connect console, create a ROMA Connect instance and select an instance edition based on service requirements.
2. **Create an integration application.**  
Create an integration application for an instance. All resources in an instance must be associated with an integration application.
3. Select ROMA Connect functions based on service scenarios.
  - **Data integration:** Connects data sources at the source and destination. A data task can be used to integrate data from the source to the destination.

- **Service integration:** Encapsulates existing backend services, data sources, and custom functions into standard RESTful APIs and exposes them to external systems.
- **Message integration:** Creates message topics through which different systems interconnect with each other to send and receive messages.
- **Device integration:** Defines device models and registers devices in the cloud. Devices are connected to the cloud through SDKs to send and receive messages.

# 2 Instances

---

[Creating a ROMA Connect Instance](#)

[Managing Instances](#)

[Restoring/Migrating Integration Assets](#)

## 2.1 Creating a ROMA Connect Instance

### 2.1.1 Preparing Required Resources

#### Overview

Before creating a ROMA Connect instance, you need to ensure availability of resources, including a virtual private cloud (VPC), subnet, and security group. Each ROMA Connect instance is deployed in a VPC and bound to specific subnets and security groups. In this way, ROMA Connect provides an isolated virtual network environment and security protection policies that can be easily configured and managed by users.

#### Required Resources

The following table lists the resources required by a ROMA Connect instance.

**Table 2-1** ROMA Connect resources

| Resource       | Requirement   | Operations  |
|----------------|---|---|
| VPC and subnet | <p>Different ROMA Connect instances can use the same or different VPCs and subnets based on site requirements. Note the following when creating a VPC and subnet:</p> <ul style="list-style-type: none"> <li>• The created VPC and ROMA Connect must be in the same region.</li> <li>• Retain the default settings unless otherwise specified.</li> </ul>   | <p>For details about how to create a VPC and subnet, see <a href="#">Creating a VPC</a>. If you need to create and use a new subnet in an existing VPC, see <a href="#">Creating a Subnet for the VPC</a>.</p>    |
| Security group | <p>Different ROMA Connect instances can use the same security group or different security groups. Note the following when creating a security group:</p> <ul style="list-style-type: none"> <li>• Set <b>Template</b> to <b>Custom</b>.</li> <li>• After a security group is created, retain the default inbound and outbound rules.</li> <li>• To use ROMA Connect, you must add the security group rules described in <a href="#">Table 2-2</a>.</li> </ul> | <p>For details about how to create a security group, see <a href="#">Creating a Security Group</a>. For details about how to add rules to a security group, see <a href="#">Adding a Security Group Rule</a>.</p> |
| (Optional) EIP | <p>If you want to access ROMA Connect through a public network, apply for an EIP. An instance needs to be bound to only one EIP.</p>  | <p>For details about how to assign an EIP, see <a href="#">Assigning an EIP</a>.</p>  |

**Table 2-2** Security group rules

| Direction | Protocol | Port | Source IP Address | Description  |
|-----------|----------|------|-------------------|--|
| Inbound   | TCP      | 80   | 0.0.0.0/0         | Access APIC through a public network (without SSL encryption). |
| Inbound   | TCP      | 443  | 0.0.0.0/0         | Access APIC through a public network (with SSL encryption).    |

| Direction | Protocol | Port | Source IP Address | Description  |
|-----------|----------|------|-------------------|--|
| Inbound   | TCP      | 1883 | 0.0.0.0/0         | Access LINK through a public network (without SSL encryption).             |
| Inbound   | TCP      | 7443 | 0.0.0.0/0         | Access LINK through a public network (using RESTful APIs).                 |
| Inbound   | TCP      | 8443 | 0.0.0.0/0         | Access LINK through a public network (with SSL encryption).                |
| Inbound   | TCP      | 9092 | 0.0.0.0/0         | Access MQS through a private network (without SASL authentication).        |
| Inbound   | TCP      | 9093 | 0.0.0.0/0         | Access MQS through a private network (with SASL authentication).           |
| Inbound   | TCP      | 9094 | 0.0.0.0/0         | Access MQS through a public network (without SASL authentication).         |
| Inbound   | TCP      | 9095 | 0.0.0.0/0         | Access MQS through a public network (with or without SASL authentication). |
| Inbound   | TCP      | 9096 | 0.0.0.0/0         | Access MQS through a public network (with or without SASL authentication). |
| Inbound   | TCP      | 9097 | 0.0.0.0/0         | Access MQS through a public network (with SASL authentication).            |
| Inbound   | TCP      | 9292 | 0.0.0.0/0         | Access MQS through a public network (using RESTful APIs).                  |

## 2.1.2 Creating an Instance

### Overview

Before using ROMA Connect, you need to create a ROMA Connect instance. A ROMA Connect instance is an independent resource space. Resources of different instances are isolated from each other. You can use one or more ROMA Connect instances as required.

 NOTE

If you delete an instance, all resource data created in the instance will be deleted. Exercise caution when performing this operation. Before deleting an instance, back up data by [exporting asset data of an instance](#).

## Prerequisites

- A VPC is available, and the subnet and security group have been configured. (For details on how to create a VPC, subnet, and security group, see [Preparing Required Resources](#).)
- The available quota of ROMA Connect instances is not 0. (If the available quota is 0, delete unnecessary instances or apply for [increasing the instance quota](#).)

## Procedure

1. Go to the [Buy Instance page](#).
2. On the **Buy ROMA Connect Instance** page, enter configuration information about the instance. The system automatically calculates the fee based on **Edition** and **Required Duration**. Then, click **Next**.

**Table 2-3** ROMA Connect instance parameters

| Parameter        | Configuration  |
|------------------|--|
| Billing Mode     | Billing mode of an instance. Select <b>Yearly/Monthly</b> .  |
| Region           | Select the region where the instance resides. Instances in different regions cannot communicate with each other. Select the nearest region to reduce network latency.  |
| AZ               | Select the availability zone (AZ) where the instance resides. Different AZs are physically isolated but can communicate with each other through a private network. <ul style="list-style-type: none"><li>• To enhance ROMA Connect availability, create instances in different AZs.</li><li>• To shorten network latency, create instances in the same AZ.</li></ul> |
| CPU Architecture | Select <b>x86</b> or <b>Kunpeng</b> based on the CPU architecture supported by the current environment.  |

| Parameter | Configuration  |
|-----------|--|
| Edition   | <p>Select the capacity specifications of the instance. The edition cannot be changed after the instance is created.</p> <ul style="list-style-type: none"><li>• Basic: 25 connections, integrating 5 to 10 systems</li><li>• Professional: 80 connections, integrating 10 to 20 systems</li><li>• Enterprise: 200 connections, integrating 20 to 30 systems</li><li>• Platinum: 800 connections, integrating over 30 systems</li></ul> <p><b>NOTE</b></p> <p>The number of connections and the number of systems are only used as a reference for selecting ROMA Connect instance edition. They do not limit the number of resources (such as integration tasks, APIs, and message topics) in an instance and are not directly associated with the resource quota of the instance.</p> <ul style="list-style-type: none"><li>• The number of systems refers to the number of users' service systems interconnecting with the ROMA Connect instance. Multiple connections can be set up between a service system and a ROMA Connect instance.</li><li>• A connection refers to an interaction between a service system and ROMA Connect. The number of connections varies depending on the functional module in ROMA Connect to connect. The mappings between the number of resources and the number of connections are as follows:<ul style="list-style-type: none"><li>• FDI: Two data integration tasks in the running state occupy one connection.</li><li>• APIC: 10 hosting APIs (not published by custom backends) occupy one connection. Five function APIs or data APIs occupy one connection.</li><li>• MQS: Three topics occupy one connection.</li><li>• LINK: 1000 devices occupy one connection.</li></ul></li></ul> <p>Example: An enterprise integrates its services using ROMA Connect.</p> <ul style="list-style-type: none"><li>• If 50 data integration tasks need to be created for data synchronization but only a maximum of 20 data integration tasks can be running concurrently, <b>the number of connections required by FDI is 10 (20/2).</b></li><li>• If 200 hosting APIs and 20 data APIs need to be created for function and data exposure, <b>the number of connections required by APIC is 24 (200/10 + 20/5).</b></li><li>• If 30 topics need to be created for message integration across systems, <b>the number of connections required by MQS is 10 (30/3).</b></li><li>• If 5000 devices need to be created for IoT management of enterprise assets, <b>the number of connections required by LINK is 5 (5000/1000).</b></li></ul> <p>In this example, the total number of connections required is 10 (FDI) +24 (APIC) +10 (MQS) +5 (LINK) = 49. Therefore, an instance of the professional edition or higher specifications should be selected.</p> |

| Parameter                     | Configuration  |
|-------------------------------|--|
| Enterprise Project            | Select an enterprise project. You can associate an instance with the enterprise project.   |
| Instance Name                 | Enter the instance name as planned.  |
| Description                   | Enter a brief description of the instance.   |
| VPC                           | Select the VPC and subnet associated with the instance. The VPC and subnet must have been created in <a href="#">Preparing Required Resources</a> .  |
| Security Group                | Select the security group associated with the instance. The security group must have been created in <a href="#">Preparing Required Resources</a> .  |
| Public Access                 | Determine whether to enable the function of accessing ROMA Connect from a public network based on service requirements.  |
| EIP                           | This parameter is available only if <b>Public Access</b> is enabled.<br>Select the elastic public network IP associated with the instance. This elastic public network IP must be applied for in advance, as described in <a href="#">Preparing Required Resources</a> .   |
| Engine Version                | Select the Kafka server version used by the instance. Currently, 1.1.0 and 2.3.0 are supported.  |
| MQS Capacity Threshold Policy | Select a processing policy that will be used when the number of messages in MQS reaches the capacity threshold (95% of the storage space). <ul style="list-style-type: none"><li>• <b>Stop creation:</b> New message creation requests will be rejected until a message is deleted according to the configured aging time. After the available storage space is greater than 5%, new message creation requests can be received. This policy applies when message retrieval processing is slow or inventory messages need to be repeatedly retrieved and cannot be deleted in advance.</li><li>• <b>Delete messages:</b> The oldest messages (10% of the total number of messages) are automatically deleted to free up storage space for new requests. This policy has no adverse impact on message creation. It applies when message retrieval is fast and there are no accumulated messages.</li></ul> |
| SASL_SSL                      | Determine whether to enable SASL transmission and SSL encryption for message transmission. Enabling this parameter better secures data transmission.<br>Enabling SASL_SSL cannot be undone.  |



| Parameter                  | Configuration  |
|----------------------------|--|
| Intra-VPC Plaintext Access | This parameter is available only if SASL_SSL is enabled. Specify whether plaintext access is used in the VPC. If intra-VPC plaintext access is enabled, SASL cannot be used to connect to MQS topics in the VPC. That is, no certificate is required for accessing topics. |
| Required Duration          | Select a validity period for the instance. The minimum duration is 1 month. If you select <b>Auto renew</b> , monthly subscriptions are renewed each month and yearly subscriptions are renewed each year.   |

3. On the **Confirm** page, select the item to confirm that you have read and agree to the customer agreement and privacy statement, and click **Pay Now** to create an instance.

The instance creation takes about 18 to 25 minutes. After the instance is created, the instance status is **Available** on the **Instances** page.

If the ROMA Connect instance fails to be created, delete the failed instance and create a new one. If the creation still fails, contact technical support.

## 2.2 Managing Instances

### 2.2.1 Viewing Details of an Instance


#### Overview


After an instance is created, you can view and edit the instance configuration information on the console, including basic information, configuration parameters, and data dictionaries, as well as import and export assets.



#### Procedure

On the **Instance Information** page of the console, click the **Basic Information** tab to view and edit basic information about the instance.

**Table 2-4** Basic information (ROMA Connect)

| Content  | Description   |
|----------|---|
| Instance | Basic information about the instance, including the instance name, instance ID, edition, description, availability zone, creation time, and enterprise project. <ul style="list-style-type: none"><li>• You can modify <b>Instance Name</b> and <b>Description</b> as required.</li><li>• Click  next to <b>Instance ID</b> to copy the instance ID.</li></ul> |

| Content               | Description  |
|-----------------------|--|
| VPC                   | VPC associated with the instance. You can click the VPC name to view the VPC configuration.  |
| Subnet                | Subnet associated with the instance. You can click the subnet name to view the subnet configuration.   |
| Security Group        | Security group associated with the instance. You can click the security group name to view the security group configuration or click <b>Edit</b> to bind a new one.  |
| Routes                | Private network segment. After a ROMA Connect instance is created, it can communicate with the VPC subnet segment specified during instance creation by default. If other private network segments need to communicate with the instance, you can configure routes by setting this parameter.  |
| Connections           | Number of instance resource connections. For details, see the <a href="#">conversion between resources and connections</a> .   |
| Billing Information   | Billing mode of the instance.  |
| Message Storage Space | Information about the MQS message storage space of the instance, including the storage class and space usage (percentage).   |
| MQS Information       | Basic MQS configuration information about an instance. Modify <b>MQS Capacity Threshold Policy</b> as required.  |
| Connection Addresses  | <p>Connection addresses of the instance:</p> <ul style="list-style-type: none"> <li>● <b>ROMA Connect Address</b></li> <li>● <b>LINK Address</b></li> <li>● <b>APIC Address</b></li> <li>● <b>APIC Intranet Address</b> (for backend and gateway components)</li> <li>● <b>FDI Intranet Egress Address</b></li> <li>● <b>MQS Intranet Address</b></li> <li>● <b>MQS Public Address</b> (available only if an EIP is bound)</li> <li>● <b>MQS Connector Address</b> (available only after you contact technical support to enable it)</li> </ul> <p>Click  next to a connection address to copy the address information.</p> |

| Content                | Description  |
|------------------------|--|
| Public Access          | EIP bound to the instance. <ul style="list-style-type: none"><li>• If an EIP has not been bound to the instance, click  on the right of the address to bind an EIP to the instance.</li><li>• If an EIP has been bound to the instance, click  next to the EIP to copy the address information.</li><li>• If an EIP has been bound to the instance, click <b>Unbind</b> on the right of the address to unbind the EIP from the instance.</li></ul> |
| Public Outbound Access | Addresses used by FDI and APIC to access public network data or services. The addresses are automatically configured during instance creation.   |

## 2.2.2 Modifying Instance Configuration Parameters

### Overview

This section describes how to configure common parameters of components in an instance. By modifying configuration parameters, you can adjust related function configurations of components.


### Constraints

Modifying instance configuration parameters will interrupt APIC. Do this during off-peak hours or when no service is running.

### Modifying Configuration Parameters

On the **Instance Information** page of the ROMA Connect console, click the **Configuration Parameters** tab and view the configuration parameters of the instance. You can also change the values of **Current Value**.

**Table 2-5** Configuration parameters

| Parameter     | Description   |
|---------------|---|
| Parameter     | Name of a parameter. You can move the cursor to  next to a parameter name to view its description. |
| Default Value | Default value of a parameter.   |
| Value Range   | Value range of a parameter.   |
| Current Value | Current value of a parameter.   |

| Parameter | Description   |
|-----------|---|
| Updated   | Time when a parameter was last updated. If the parameter has never been modified, this parameter is left blank. |
| Operation | Click <b>Edit</b> to change the value of <b>Current Value</b> .   |

## Parameter Description

The following table lists the instance configuration parameters of ROMA Connect.

**Table 2-6** Instance configuration parameters

| Parameter            | Description  |
|----------------------|--|
| ratelimit_api_limits | Default request throttling value applied to all APIs. If no request throttling policy is bound to an API, the total number of times the API can be called is determined by this parameter value. If a request throttling policy has been bound to an API, the total number of times the API can be called is determined by the bound policy.   |
| request_body_size    | Maximum size of the body allowed in an API request.  |
| backend_timeout      | Maximum timeout duration for ROMA Connect to send a request to a backend service.  |
| app_token            | Whether to enable app_token authentication. After this function is enabled, the obtained access token can be used in API requests for authentication during API calling. <ul style="list-style-type: none"><li>• <b>app_token_expire_time</b> indicates the validity period of the access token. Before the access token expires, you must obtain a new access token.</li><li>• <b>refresh_token_expire_time</b> indicates the validity period of the refresh token. A refresh token is used to obtain a new access token.</li><li>• <b>app_token_uri</b> indicates the URI used for obtaining an access token.</li><li>• <b>app_token_key</b> indicates the encryption key of the access token.</li></ul> |
| app_api_key          | Whether to enable app_api_key authentication. After this function is enabled, you can add the <b>apikey</b> parameter to an API request to carry the key of an integration application (or the AppKey of a client) for authentication during API calling.  |

| Parameter  | Description   |
|------------|---|
| app_basic  | Whether to enable app_basic authentication. After this function is enabled, you can add the <b>Authorization</b> parameter to an API request to carry the key and secret of an integration application (or the AppKey and AppSecret of a client) for authentication during API calling.   |
| app_jwt    | Whether to enable app_jwt authentication. After this function is enabled, you can add the <b>Authorization</b> and <b>Timestamp</b> parameters to the API request to carry the key and secret (or AppKey and AppSecret of the client) of the integration application and the timestamp for authentication during API calling.<br><br><b>app_jwt_auth_header</b> indicates the app_jwt authentication header, that is, the Header parameter of app_jwt authentication information carried in an API request. The default value is <b>Authorization</b> . |
| app_secret | Whether to enable app_secret authentication. After this function is enabled, you can add the <b>X-HW-ID</b> and <b>X-HW-AppKey</b> parameters to an API request to carry the key and secret of an integration application (or the AppKey and AppSecret of a client) for authentication during API calling.  |
| public_key | Whether to enable the backend signature of the public_key type. After this function is enabled, signatures of the public_key type can be used in backend signature authentication.<br><br>public_key_uri_prefix indicates the URI prefix used to obtain the secret corresponding to public_key. The URI format is as follows: <code>https://{APIC connection address}{public_key_uri_prefix}{public_key signature key name}</code> .  |
| app_route  | Whether to support IP address access. After this function is enabled, APIs in non-DEFAULT groups can be called by using IP addresses.   |
| cascade    | Whether to enable the API cascading function. After this function is enabled, APIs can be called across instances. <ul style="list-style-type: none"> <li>• cascade_auth_key indicates the encryption key used for authentication between APIs in the cascading relationship.</li> <li>• cascade_instance_ids indicates the ID list of cascading instances. Only instances specified by this parameter can establish the cascading relationship with the current instance.</li> </ul>   |

| Parameter                   | Description  |
|-----------------------------|--|
| default_group_hide          | Whether to hide the DEFAULT group. After this function is enabled, the DEFAULT group is hidden.  |
| livedata_config             | <p>Custom backend configuration.</p> <ul style="list-style-type: none"> <li>• <b>sandbox_max_memory</b> indicates the maximum memory required for executing a script of a function backend.</li> <li>• <b>sandbox_max_cpu_time</b> indicates the maximum CPU time required for executing a script of a function backend.</li> <li>• <b>livedata_env</b> indicates the running mode of a custom backend. If this parameter is set to <b>online</b>, the custom backend will return responses in the online format.</li> <li>• <b>gw_address_protocol</b> indicates the default request protocol transferred through the <b>DICT:gw_rest_float_addr</b> field in a function script.</li> <li>• <b>procedure_async</b> indicates whether the stored procedure is executed in asynchronous mode.</li> <li>• <b>dataapi_return_type</b> indicates the return format of a data backend.</li> </ul> |
| backend_client_certificate  | Whether to enable two-way authentication for a backend. After this function is enabled, you can configure the two-way authentication for a backend when configuring the backend information during API creation.   |
| ssl_ciphers                 | HTTPS cipher suite that can be configured. You can select the cipher suites as required.   |
| apiclient_first_use_x_hw_id | Whether to preferentially use the <b>X-HW-ID</b> field to verify the ApiClient class of the custom backend.  |

| Parameter             | Description  |
|-----------------------|--|
| real_ip_from_xff      | <p>Whether to use the IP addresses in the <b>X-Forwarded-For</b> header as the criterion for access control and request throttling.</p> <p><b>xff_index</b>: Sequence number of an IP address in the <b>X-Forwarded-For</b> header. The value can be positive, negative, or 0.</p> <ul style="list-style-type: none"><li>• If the value is 0 or positive, obtain the IP address of the corresponding index in the <b>X-Forwarded-For</b> header.</li><li>• If the value is negative, obtain the IP address of the indicated reverse sequence in the <b>X-Forwarded-For</b> header.</li></ul> <p>For example, assume that the <b>X-Forwarded-For</b> header of a request received by API gateway contains three IP addresses: IP1, IP2, and IP3. If the value of <b>xff_index</b> is 0, IP1 is obtained. If the value is 1, IP2 is obtained. If the value is -1, IP3 is obtained. If the value is -2, IP2 is obtained.</p>                      |
| custom_log            | <p>Whether to enable custom logs. After the custom log function is enabled, the specified location (header, query, and cookie) and parameter values are printed in the <b>calling logs</b> of all APIs in the ROMA Connect instance.</p> <p>After this function is enabled, click <b>Add</b> to add the parameters to be printed in the calling logs.</p> <p><b>NOTE</b></p> <ul style="list-style-type: none"><li>• Custom logs print only the content of requests initiated from clients and do not print the constants and system parameters defined in APIC.</li><li>• Custom logs support up to 10 parameters. The total size of all parameter fields cannot exceed 2 KB.</li><li>• Some special characters in parameter values printed by custom logs will be encoded. For example, a plus sign (+) will be encoded as a space, double quotation marks (") encoded as <b>\x22</b>, and a backslash (\) encoded as <b>\x5C</b>.</li></ul> |
| real_ip_header_getter | <p><b>header_getter</b>: Whether to obtain source IP addresses from custom headers for access control and request throttling policies to take effect.</p>  |
| vpc_name_modifiable   | <p>Whether to allow load balance channel name modification. When this function is enabled, you can modify the name of load balance channels. However, the <b>VPC Channel Management - Project-Level</b> API cannot be called currently.</p>  |

| Parameter                                 | Description  |
|---|--|
| default_group_host_trustlist              | <p>Whether to allow access to APIs in the default group from the IP addresses that are not inbound access addresses of the current instance. When this function is enabled, IP addresses can be added to access APIs in the default group.</p> <ul style="list-style-type: none"><li>• <b>Default Group API Access from Custom IP Addresses:</b> whether APIs in the default group can be accessed from custom IP addresses.</li><li>• <b>IP Addresses:</b> custom IP addresses to access the APIs in the default group. Separate multiple IP addresses with semicolons (;).</li></ul> |
| data_api_column_types_converted_to_string | <p>Whether to allow data backends to support converting data column types to String. When this function is enabled, a selected data column type can be converted to String.</p> <ul style="list-style-type: none"><li>• <b>Configure Data Column Types:</b> whether to allow data backend column types to be converted to String.</li><li>• <b>Types:</b> data column types to be converted to String (only NVARCHAR2 is supported currently).</li></ul>   |
| sse_strategy                              | <p>Whether to enable Server-Sent Events (SSE) transmission. It is disabled by default. Once enabled, the responses of backend APIs are output in streaming mode for character-based rendering.</p> <p><b>NOTICE</b><br/>The sse_strategy configuration can be modified 1 minute after being completed.</p>   |
| request_custom_config                     | <p>Configure client request parameters.</p> <ul style="list-style-type: none"><li>• <b>HTTP/2:</b> Enabled by default.</li><li>• <b>request_body_timeout:</b> Timeout for client request body. Default: 8s. Modify this parameter if the network condition is poor or the request body is too large.</li></ul> <p><b>NOTICE</b><br/>The client request configuration can be modified 1 minute after being completed.</p>   |

## 2.2.3 Creating a Data Dictionary

### Overview

A data dictionary is used to open up key values in a system database to users. **DEVICE\_TYPE** is the default data dictionary. You can also customize other data dictionaries as required.




## Procedure

1. On the **Instance Information** page of the console, click the **Data Dictionaries** tab.
2. Click **Create Data Dictionary**.
3. In the **Create Data Dictionary** dialog box, set dictionary parameters and click **OK**.

**Table 2-7** Parameters for creating a data dictionary

| Parameter        | Description   |
|------------------|---|
| Dictionary Code  | Enter the unique ID of a dictionary.                                    |
| Dictionary Name  | Enter a dictionary name.  |
| Extended Field 1 | These parameters are reserved for the dictionary and can be left empty. |
| Extended Field 2 |   |
| Description      | Enter a brief description of the dictionary.                            |

4. After the data dictionary is created, click  next to the dictionary name to expand the dictionary item list.
5. Click **Create Dictionary Item**.
6. In the **Create Dictionary Item** dialog box, set dictionary item parameters and click **OK**.

**Table 2-8** Parameters for creating a dictionary item

| Parameter        | Description  |
|------------------|--|
| Item Code        | Enter the unique ID of a dictionary item.                                    |
| Item Name        | Enter the dictionary item name.  |
| Extended Field 1 | These parameters are reserved for the dictionary item and can be left empty. |
| Extended Field 2 |  |
| Description      | Enter a brief description of the dictionary item.                            |

## 2.2.4 Importing and Exporting Assets

### Overview

ROMA Connect allows you to export application and task assets of an instance as files to a local PC or import local asset files to ROMA Connect to migrate assets in batches.

## Prerequisites

- Only integration asset data can be imported, excluding asset-related instance configuration data. When you have imported integration assets, configure asset-related instance data. For details, see [Restoring/Migrating Integration Assets](#).
- Ensure that the instance configurations during import and export are the same. Otherwise, the import may fail. For example, if `app_route` is not enabled on the exported instance but is enabled on the instance to be imported, the asset import may fail when the imported asset package contains APIs with conflict paths.
- Request throttling policies, access control policies, and signature keys of APIs cannot be imported or exported.
- If the imported assets contain APIs using FunctionGraph backends, ensure that the FunctionGraph version/alias of the current user is the same as that in the asset package.

## Importing Assets

1. On the **Instance Information** page of the console, click **Import Asset** in the upper right corner.
2. In the **Import Asset** dialog box, select **Append** or **Overwrite** based on the site requirements.

The data source IDs of different instances must be different. If the data source IDs are the same, select **Overwrite**.

3. If you select **Overwrite**, set whether to override the environment configurations.
  - **Override**: The imported data source will overwrite the existing data source with the same name.
  - **Not Override**: The imported data source will not overwrite the existing data source with the same name.

4. Enter the prefix  
FDI tasks in the same instance must have different names. To import FDI tasks with same name, enter a prefix for each task before importing

5. In the dialog box displayed, select a local asset file and import it.
6. After the import is complete, click **Close**. You can view the imported assets in the instance.

### NOTE

- The custom authorizer name is globally unique. If a custom authorizer with the same name already exists in the instance when the asset is imported, random characters will be added to the end of the imported custom authorizer name to avoid duplicate names.
- If the following error information is displayed during asset import, contact technical support to upgrade the instance:

```
"CipherEntry":{"reason":{"resourceType":"cipherEntries"},"resourceId\n":"\\","resourceName":null,"errorCode":"APIG.9999","errorMsg":"System error\n"}}
```

## Exporting Assets

1. On the **Instance Information** page of the console, click **Export Asset** in the upper right corner.
2. Export assets.
  - Exporting specified assets: Select the application or task assets to be exported and click **Export Selected** to export the asset files to the local PC.
  - Exporting all assets: Click **Export All** to export all asset files to a local directory.

If you need to export all API asset information when exporting application assets, select **Export all API assets in applications**. All API asset information including self-created and authorized APIs will be exported. If you do not select this parameter, information about self-created and authorized APIs will be exported.

3. After the export is complete, click **Close**. You can view the exported assets in the local PC.

## 2.3 Restoring/Migrating Integration Assets

### 2.3.1 Overview

#### Scenarios

ROMA Connect allows you to export applications and tasks of an instance as an asset file and import an asset file to restore or migrate integration assets. For details on how to import and export assets, see [Importing and Exporting Assets](#).

- **Asset restoration:** You can export integration assets from a ROMA Connect instance for data backup. In the event of a faulty instance or data loss, you can import the backup asset file to the instance for restoration.  
You are advised to periodically export asset files (for example, on a monthly basis) for data backup and restoration.
- **Asset migration:** You can import an integration asset file that was exported from existing ROMA Connect instances into a new instance to quickly deploy and reuse existing integration assets.

Only integration asset data can be imported, excluding asset-related instance configuration data. After integration asset import, you need to reconfigure asset-related instance data.

**Table 2-9** Instance configurations

| Category                    | Item   | Description  | Processing Method  |
|-----------------------------|--|--|--|
| Public instance information | Integration application secret                               | The secrets of integration applications are non-asset data.  | After assets are imported, ROMA Connect automatically generates new secrets. Notify interworking service applications to update the secrets accordingly. |
|                             | Integration application authorization information            | The authorization information of integration applications is non-asset data.   | Reconfigure user authorization after assets are imported.  |
| Data integration            | Data source connection information                           | Includes the addresses, accounts, and passwords, which are non-asset data.   | Configure connection information for the corresponding data sources after assets are imported.   |
| Service integration         | APIs that have not been published in the RELEASE environment | Include APIs that have not been published and APIs that have been published in a non-RELEASE environment. These APIs are non-asset data.   | Re-create and publish APIs as required after assets are imported.  |
|                             | Environment information of APIs                              | Includes the independent domain names, environment variables, cloud server information of load balance channels, and non-default environments. This information is non-asset data. | Reconfigure environment information after assets are imported.   |
|                             | Control policy information of APIs                           | Includes request throttling policies, access control policies, client access control policies, and client quota policies. This information is non-asset data.                      | Reconfigure control policy information after assets are imported.  |

| Category           | Item  | Description  | Processing Method   |
|--------------------|---|--|---|
|                    | Secret information of APIs                            | Includes signature keys, client's AppCodes, and custom backends' configurations of the password or certificate type. This information is non-asset data. | After asset import: <ul style="list-style-type: none"> <li>ROMA Connect automatically generates new keys and secrets for signature keys. Notify backend services to update the keys and secrets accordingly.</li> <li>Reconfigure the client's AppCode.</li> <li>Reconfigure the configurations of the password or certificate type for the custom backends.</li> </ul> |
|                    | Blacklist and whitelist specific to API authorization | The blacklist and whitelist specific to API authorization is non-asset data.   | Reconfigure blacklist and whitelist information after assets are imported.  |
| Device integration | Device information                                    | The device information is non-asset data.  | Re-create devices after assets are imported.  |
|                    | Device rule information                               | The device rule information is non-asset data.   | Re-create device rules after assets are imported.   |

## Restoration/Migration Process

The following table describes the restoration/migration process of integration assets.

**Table 2-10** Service process

| No. | Operation                                | Description   |
|-----|--|---|
| 1   | <b>Preparations</b>                      | Obtain an integration asset package and collect information about interconnection with peripheral systems (such as data sources, backend services, and service applications). |
| 2   | <b>Modifying Instance Configurations</b> | Modify the security group and configuration parameters of the instance you want to restore or a new instance.   |

| No. | Operation  | Description   |
|-----|--|---|
| 3   | <b>Importing Integration Assets</b>                        | Import the integration asset package to the instance you want to restore or a new instance.     |
| 4   | <b>Modifying Integration Application Configurations</b>    | Modify the authorization configurations of integration applications.                            |
| 5   | <b>Modifying Data Source Configurations</b>                | Modify the connection configuration of data sources.  |
| 6   | <b>Modifying API Configurations</b>                        | Modify the environment, control policy, secret, and blacklist/whitelist configurations of APIs. |
| 7   | <b>Modifying Device Configurations</b>                     | Create devices and rules in batches.  |
| 8   | <b>Service Interconnection Adaptation and Verification</b> | Interconnect with peripheral systems and verify services.                                       |

## 2.3.2 Preparations

Before you restore or migrate integration assets, make the following preparations:

- Prepare an integration asset package.  
Obtain the integration asset package you want to import.
- Obtain configuration information of the source instance to which an asset package belongs. This information is required only in asset migration scenarios.

The configuration information includes the security group configuration, instance configuration parameters, and data dictionary information of instances.

- Obtain interconnection information of peripheral systems. The information is required only in asset migration scenarios.

| System              | Item   | Purpose   |
|---------------------|--|---|
| Data source         | IP address, username, and password for accessing a data source | Used for ROMA Connect to connect to the data source         |
| Service application | Network segment of a service application                       | Used for ROMA Connect to connect to the service application |

## 2.3.3 Modifying Instance Configurations

### Overview

Before you import integration assets, modify the parameter configurations of the instance to which you want to import integration assets, including the security group configuration, instance configuration parameters, and data dictionaries.

In asset migration scenarios, the instance configuration parameters must be the same as those of the source instance to which the asset package belongs. Otherwise, the assets cannot be imported.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. Modify the security group configuration. This step is performed only in asset migration scenarios. Otherwise, skip this step.
  - a. On the **Basic Information** tab page of the **Instance Information** page, click the security group name.
  - b. On the security group configuration page, modify the security group rule based on the source instance information obtained in [Preparations](#) and the actual networking requirements of the new instance.  
For details about the security group rule of ROMA Connect instances, see [Table 2-2](#).
3. Modify instance configuration parameters. This step is performed only in asset migration scenarios. Otherwise, skip this step.  
On the **Instance Information** page, click the **Configuration Parameters** tab and modify the configuration parameters based on the source instance information obtained in [Preparations](#).
4. Add data dictionaries.  
On the **Instance Information** page, click the **Data Dictionaries** tab and add data dictionaries based on the source instance information obtained in [Preparations](#).  
For details about how to create a data dictionary, see [Creating a Data Dictionary](#).

## 2.3.4 Importing Integration Assets

### Overview

You can use the asset import function to quickly import an integration asset package to the ROMA Connect instance.

### Prerequisites

- You have obtained an integration asset package.
- In asset migration scenarios, you have modified instance configuration parameters. For details, see [3](#).

## Procedure

1. On the **Instances** page of the ROMA Connect console, click **Import Asset** in the upper right corner.
2. In the **Import Asset** dialog box, select **Append** or **Overwrite** based on the site requirements.
  - In asset restoration scenarios, select **Overwrite**.
  - In asset migration scenarios, select **Append**.
3. In the dialog box displayed, select a local asset file and import it.
4. After the import is complete, click **Close**. You can view the imported assets in the instance.

## 2.3.5 Modifying Integration Application Configurations

### Overview

User authorization information of integration applications is non-asset data, which must be reconfigured after you import integration assets.

- In asset restoration scenarios, you need to configure the imported integration applications.
- In asset migration scenarios, you can configure the integration applications based on the site requirements.

### Prerequisites

You have imported integration assets. For details, see [Importing Integration Assets](#).

### Procedure

1. On the **Integration Applications** page of the ROMA Connect instance console, click **Application Authorization Management** on the right of an integration application.
2. In the dialog box displayed, configure authorization information for the application.

For details about how to configure authorization information, see [Configuring Integration Application Authorization](#).
3. Click **OK**.

## 2.3.6 Modifying Data Source Configurations

### Overview

Data source connection information includes the connection address, account, and password, which is non-asset data. After you complete importing integration assets, you must reconfigure the data source connection information.



## Prerequisites

You have imported integration assets. For details, see [Importing Integration Assets](#).

## Procedure

1. On the **Data Sources** page of the ROMA Connect console, click **Edit** on the right of the target data source.
2. On the **Edit Data Source** page, modify connection information about the data source.
  - In asset restoration scenarios, change the password.
  - In asset migration scenarios, change the IP address, username, and password of the data source.

For details about the data source parameters, see [Connecting to Data Sources](#).

3. After the modification is complete, click **Check Connectivity** to verify the connectivity between ROMA Connect and the data source.
4. Click **Save**.

## 2.3.7 Modifying API Configurations

### Overview

The environment information, control policy information, secret information, blacklist and whitelist for API authorization, and APIs that are not published in the RELEASE environment are not asset data. You need to reconfigure this information after importing integration assets.

- Environment information includes the independent domain names, environment variables, cloud server information of load balance channels, and non-default environments.
- Control policy information includes request throttling policies, access control policies, client access control policies, and client quota policies.
- Secret information includes clients' AppCodes and custom backends' configurations of the password or certificate type.

### Prerequisites

You have imported integration assets. For details, see [Importing Integration Assets](#).

### Configuring Environment Information

1. Bind an independent domain name.
  - a. On the instance console, choose **API Connect > API Groups**. Click the name of an API group to view details.
  - b. Choose **Group Information** and click **Bind Independent Domain Name** in the **Independent Domain Names** area to bind an independent domain name to the API group.

- For details about the parameters, see [Binding Domain Names](#).
- c. Click **OK**.
2. Create an environment.
    - a. On the instance console, choose **API Connect > API Policies**, and click the **Environments** tab.
    - b. Click **Create** and create an environment for publishing APIs.
    - c. Click **OK**.
  3. Add environment variables.
    - a. On the instance console, choose **API Connect > API Groups**. Click the name of an API group to view details.
    - b. Click the **Group Information** tab in the upper left corner. In the **Environment Variables** area, select the environment to which you want to add variables, and click **Add Environment Variable**.

For details about the parameters, see [Creating an Environment Variable](#).
    - c. Click **OK**.
  4. Modify a load balance channel server.
    - a. On the instance console, choose **API Connect > API Policies**, and click the **Load Balance Channels** tab.
    - b. Click **Modify** on the right of a load balance channel.
    - c. Modify the server configuration.
      - For channels whose mode is set to **Select cloud servers**, click **Add Cloud Server** to add servers.
      - For channels whose mode is set to **Specify IP addresses**, modify the backend service addresses as required.

For details about the parameters, see [\(Optional\) Creating a Load Balance Channel](#).
    - d. After the configuration is complete, click **Finish**.

## Configuring Control Policy Information

1. Configure a request throttling policy.
  - a. On the instance console, choose **API Connect > API Policies**, and click the **Policies** tab.
  - b. Click **Create Policy**. On the **Select Policy Type** page, select **Request Throttling** in the **Traditional Policies** area.

For details about the parameters, see [Creating a Request Throttling Policy](#).
  - c. Click **OK**.
  - d. Bind the request throttling policy to the API, application, or tenant by following the procedure described in [Configuring API Request Throttling](#).
2. Configure an access control policy.

- a. On the instance console, choose **API Connect > API Policies**, and click the **Policies** tab.
  - b. Click **Create Policy**. On the **Select Policy Type** page, select **Access Control** in the **Traditional Policies** area.  
For details about the parameters, see [Creating an Access Control Policy](#).
  - c. Click **OK**.
  - d. Bind an access control policy to the API by following the procedure described in [Configuring API Access Control](#).
3. Configure a credential access control policy.
    - a. On the instance console, choose **API Connect > Credentials**, and click the **Credentials** tab.
    - b. Click **Set Access Control** on the right of a credential.  
For details about the parameters, see [Configuring API Credential Access Control](#).
    - c. Click **OK**.
  4. Configure a credential quota policy.
    - a. On the instance console, choose **API Connect > Credentials**, and click the **Credential Quota Policies** tab.
    - b. Click **Create Credential Quota Policy**.  
For details about the parameters, see [Creating a Credential Quota Policy](#).
    - c. Click **OK**.
    - d. Bind the quota policy to a credential by following the procedure described in [Configuring API Credential Quotas](#).

## Configuring Secret Information

1. Add a credential for simple authentication.
  - a. On the instance console, choose **API Connect > Credentials**, and click the **Credentials** tab.
  - b. Click the name of a credential.
  - c. In the **AppCode** area, click **Add AppCode** to add an AppCode for the credential.
  - d. Click **OK**.
2. Add the configurations of the password or certificate type for custom backends.
  - a. On the instance console, choose **API Connect > Custom Backends**, and click the **Configurations** tab.
  - b. Click **Modify** on the right of a configuration item.
    - For configuration of the password type, change the password.
    - For configuration of the certificate type, modify the certificate content and private key.
  - c. Click **OK**.

## Modifying Blacklist and Whitelist Configurations of API Authorization

Only APIs that are accessed using App authentication support authorization configuration.

1. On the instance console, choose **API Connect > APIs**. Choose **More > Authorize Credentials** of an API.
2. On the page displayed, click **Enable Green Channel > Modify** on the right of an authorized credential to modify the blacklist and whitelist.
3. Click **OK**.

## Creating an API and Publishing It in a Non-RELEASE Environment

1. On the instance console, choose **API Connect > APIs** and click **Create API**. For details about the parameters, see [Creating an API](#).
2. Publish the API to an environment other than RELEASE by following the procedure described in [Publishing an API](#).

## 2.3.8 Modifying Device Configurations

### Overview

Device and device rule information is not asset data. Reconfigure the information after importing integration assets. You can use the import function to restore or create devices and rules in batches.

### Prerequisites

You have imported integration assets. For details, see [Importing Integration Assets](#). Ensure that integration applications and products to which devices and rules belong have been successfully imported.

### Procedure

1. (Optional) Prepare one device file and one rule file.  
Obtain the exported device file and rule file from the source instance to which the asset package belongs.
2. Create a device.
  - a. On the ROMA Connect console, choose **LINK > Device Management**.
    - If you have obtained a device file, click **Import** to create a device by importing the file.
    - If you have not obtained a device file, click **Create Device** to manually create a device.  
For details about the parameters, see [Creating a Device](#).
  - b. After the import is successful, you can view the imported device in the device list.
3. Create a rule.
  - a. On the ROMA Connect console, choose **LINK > Rule Engine**.

- If you have obtained a rule file, click **Import** to create a rule by importing the file.
  - If you have not obtained a rule file, click **Create Rule** to manually create a rule.  
For details about the parameters, see [Creating a Rule](#).
- b. After the import is successful, you can view the imported rule in the rule list.

## 2.3.9 Service Interconnection Adaptation and Verification

### Overview

After you have restored or migrated integration assets in the ROMA Connect instance, notify the service applications interworking with ROMA Connect to update their interconnection configurations so that they can process services properly.

### Service Interconnection Adaptation

The following table describes the interconnection configurations to update on the service applications or devices connected to ROMA Connect.

| Category            | Service Scenario                           | Configuration Data   |
|---------------------|--|--|
| Service integration | Calling an API                             | <ul style="list-style-type: none"><li>• API calling address (required only in asset migration scenarios)</li><li>• API authentication information, including the keys, secrets, AppCodes, and AK/SK of integration applications/clients</li><li>• Key and secret of backend signature keys</li></ul> |
| Message integration | Sending and receiving messages             | <ul style="list-style-type: none"><li>• Topic connection address (required only in asset migration scenarios)</li><li>• Key and secret used when SASL authentication is enabled.</li></ul>   |
| Device integration  | Reporting messages and delivering commands | <ul style="list-style-type: none"><li>• Device connection address (required only in asset migration scenarios)</li><li>• Access authentication information of devices, including client IDs, usernames, and passwords</li></ul>  |

## Service Verification

After you update the interconnection configurations, check whether service applications can process services properly.

- Check whether a service application can call APIs.
- Check whether a service application can send messages to and receive messages from topics of ROMA Connect.
- Check whether a device can send data to and receive data from ROMA Connect and whether the rule engine can forward data.

# 3 Integration Application Management

[Creating an Integration Application](#)

[Configuring Integration Application Authorization](#)

## 3.1 Creating an Integration Application

### Overview

ROMA Connect uses integration applications to isolate resources of different users in the same instance. Resources (such as data sources, APIs, topics, and products) created in a ROMA Connect instance must belong to an integration application. By default, IAM users without the **Tenant Administrator** permission can view and manage only the integration applications and resources created by themselves, while accounts assigned the permission can view and manage all resources created by their IAM users.

To share resources created by yourself to other IAM users, you can authorize the integration applications to other IAM users. For details, see [Configuring Integration Application Authorization](#).

### Creating an Integration Application

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane, choose **Integration Applications**. In the upper right corner of the page, click **Create Integration Application**.
3. In the dialog box displayed, enter the application information and click **OK**.

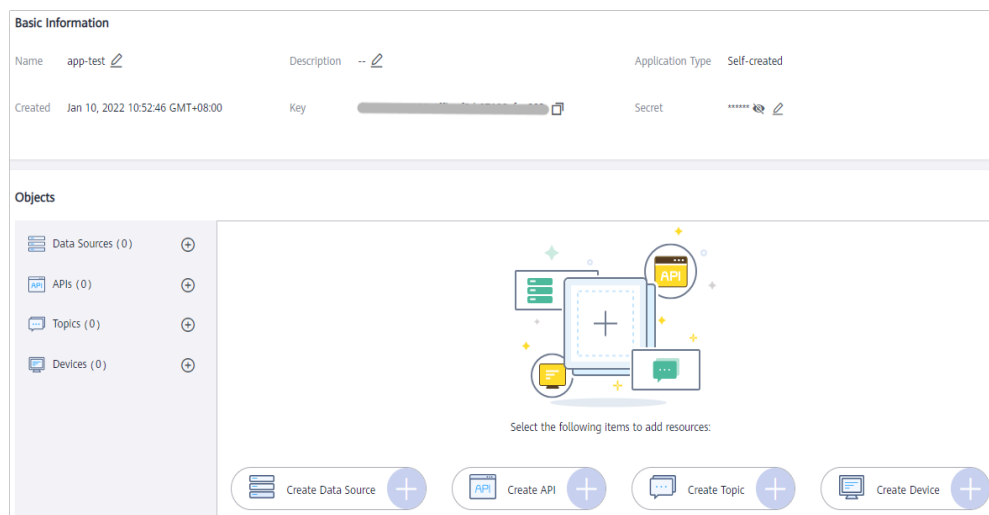
| Parameter   | Description   |
|-------------|---|
| Name        | Set an application name as required.                                  |
| Description | Enter the description of the application. This parameter is optional. |

| Parameter | Description   |
|-----------|---|
| Key       | Enter the key of the integration application. If this parameter is not specified, the system automatically generates a key.       |
| Secret    | Enter the secret of the integration application. If this parameter is not specified, the system automatically generates a secret. |

## Viewing and Editing an Integration Application


On the **Integration Applications** page of the ROMA Connect console, click the integration application name to view its details.

**Figure 3-1** Viewing application details





**Table 3-1** Integration application information

| Content           | Description  |
|-------------------|--|
| Basic Information | <p>Basic parameters of the application, including <b>Name</b>, <b>Description</b>, <b>Integration Application Type</b>, <b>Created</b>, <b>Key</b>, and <b>Secret</b>. <b>Key</b> and <b>Secret</b> can be used to authorize the access to application resources.</p> <ul style="list-style-type: none"><li>You can modify <b>Name</b> and <b>Description</b> as needed.</li><li>You can click  on the right of <b>Secret</b> to view secret information.</li><li>You can reset the secret when required. After the secret is reset, the old secret will automatically become invalid.</li><li>For accounts, the integration applications are all self-created applications. For IAM users, the integration applications created by themselves are called self-created applications, and the integration applications created and authorized by other IAM users are called authorized applications.</li></ul> |
| Objects           | <p>View, create, edit, and delete resources in an integration application, including <b>data sources</b>, <b>APIs</b>, <b>topics</b>, and <b>devices</b>.</p>  |

## 3.2 Configuring Integration Application Authorization

### Overview

ROMA Connect provides strict permissions management for user resources. In one instance, IAM users without the **Tenant Administrator** permission can view and manage only the integration applications and resources created by themselves. With integration application authorization, IAM users can share applications and resources with other IAM users under the same account.

#### NOTE

This function allows permissions sharing among users under the same account. Operations permissions on application objects are still managed through IAM.


### Configuring Integration Application Authorization

- Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
- In the navigation pane on the left, choose **Integration Applications**. On the page displayed, locate the target integration application and click **Application Authorization Management**.

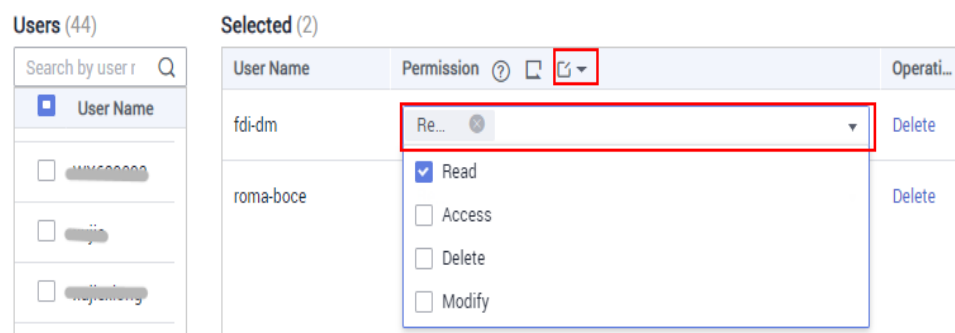
**NOTE**

An administrator (account with the **Tenant Administrator** permission) can authorize integration applications created by all its IAM users, but a common user without the permission can authorize only the integration applications they create.

3. In the dialog box displayed, grant permissions to IAM users.
  - In the **Users** area, select the IAM users to be authorized.
  - In the **Selected** area, configure management permissions of the integration application for selected IAM users.

You can grant permissions to each user separately or click  on the right of **Permission** to grant permissions to all selected users.


**Figure 3-2** Configuring application permissions



**Table 3-2** Application permissions

| Permission | FDI   | APIC   | MQS                                     | LINK  |
|------------|---|--|---|---|
| read       | View data sources of applications.            | View, debug, and export APIs of applications.                              | View and export topics of applications. | View and export devices, products, and rules of applications, as well as debug devices.                               |
| modify     | Create and edit data sources of applications. | Create, edit, release, take APIs offline, and import APIs of applications. | Create and edit topics of applications. | Create, edit, and import devices, products, and rules of applications, as well as reset device and product passwords. |

| Permission | FDI                                    | APIC  | MQS   | LINK   |
|------------|--|---|---|--|
| delete     | Delete data sources of applications.   | Delete APIs of applications.  | Delete topics of applications.                    | Delete devices, products, and rules of applications, product properties, device topics, as well as rule data sources and destinations.   |
| access     | N/A                                    | Configure authorization , access control, request throttling, and signature key binding for APIs of applications. | Configure permissions for topics of applications. | Deliver commands to and forcibly take offline devices, as well as configure plug-ins for devices that use the OPC UA or Modbus protocol. |
| admin      | Application administrator permissions. |   |   |  |

4. Click **OK**.  
In the integration application list, click  of an integration application to view its authorized IAM users and the permissions assigned.
5. Log in to a ROMA Connect console as the IAM user authorized in [3](#) and access the corresponding instance to check whether the authorization is successful.

# 4 Data Source Management

---

[Data Sources Supported by ROMA Connect](#)

[Connecting to an API Data Source](#)

[Connecting to an ActiveMQ Data Source](#)

[Connecting to an ArtemisMQ Data Source](#)

[Connecting to a DB2 Data Source](#)

[Connecting to a DIS Data Source](#)

[Connecting to a DWS Data Source](#)

[Connecting to the DM Data Source](#)

[Connecting to a Gauss100 Data Source](#)

[Connecting to an FTP Data Source](#)

[Connecting to an HL7 Data Source](#)

[Connecting to a HANA Data Source](#)

[Connecting to a HIVE Data Source](#)

[Connecting to an LDAP Data Source](#)

[Connecting to an IBM MQ Data Source](#)

[Connecting to a Kafka Data Source](#)

[Connecting to a MySQL Data Source](#)

[Connecting to a MongoDB Data Source](#)

[Connecting to an MQS Data Source](#)

[Connecting to an MRS Hive Data Source](#)

[Connecting to an MRS HDFS Data Source](#)

[Connecting to an MRS HBase Data Source](#)

[Connecting to an MRS Kafka Data Source](#)

[Connecting to an OBS Data Source](#)  
[Connecting to an Oracle Data Source](#)  
[Connecting to a PostgreSQL Data Source](#)  
[Connecting to a Redis Data Source](#)  
[Connecting to a RabbitMQ Data Source](#)  
[Connecting to a RocketMQ Data Source](#)  
[Connecting to an SAP Data Source](#)  
[Connecting to an SNMP Data Source](#)  
[Connecting to a SQL Server Data Source](#)  
[Connecting to a GaussDB\(for MySQL\) Data Source](#)  
[Connecting to a WebSocket Data Source](#)  
[Connecting to a Custom Data Source](#)

## 4.1 Data Sources Supported by ROMA Connect

### Data Sources Supported by Data Integration Task

 NOTE

- [Table 4-1](#) lists all the data source types for data integration. For details about the task types and integration modes supported by each data source, see [Connecting to Data Sources](#).
- Users who connect to a data source must have permission to create a session for connecting to the database.

**Table 4-1** Data sources supported for data integration

| Data Source Type | Compatibility   |
|------------------|---|
| API              | N/A   |
| ActiveMQ         | 5.15.9  |
| ArtemisMQ        | 2.9.0   |
| DB2              | 9.7   |
| DIS              | N/A   |
| DWS              | 1.3.4   |
| DM               | -   |
| FTP              | N/A   |
| Gauss100         | FusionInsight_LibrA_V100R003C20 and FusionInsight_LibrA_V300R001C00 |

| Data Source Type    | Compatibility  |
|---------------------|--|
| HL7                 | 2.1, 2.2, 2.3, 2.3.1, 2.4, 2.5, 2.6, 2.7, 2.8, or 2.8.1          |
| HANA                | 1.0  |
| IBM MQ              | 9.1  |
| Kafka               | 1.1.0 and 2.3.0  |
| LDAP                | N/A  |
| MongoDB             | 3.4  |
| MQS                 | N/A  |
| MRS HBase           | MRS 3.**   |
| MRS HDFS            | MRS 3.**   |
| MRS Hive            | MRS 3.**   |
| MRS Kafka           | MRS 3.**   |
| MySQL               | 5.7, 8.0   |
| OBS                 | 3  |
| Oracle              | 11.2g (not recommended), 12.1g (not recommended), 12.2g, and 19c |
| PostgreSQL          | 11   |
| RabbitMQ            | 3.6.10   |
| RocketMQ            | 4.7.0  |
| Redis               | 3.0.7, 4.0.11  |
| SAP                 | SAP Java Connector 3.0.19  |
| SNMP                | v1, v2, v3   |
| SQL Server          | 2014, 2019, and 2022   |
| GaussDB(for MySQL)  | 2.0.15.6   |
| WebSocket           | N/A  |
| Custom data sources | N/A  |

## Data Sources Supported by Data APIs

**Table 4-2** Data sources supported by data APIs

| Data Source Type | Compatibility   |
|------------------|---|
| Gauss100         | FusionInsight_LibrA_V100R003C20 and FusionInsight_LibrA_V300R001C00 |
| MySQL            | 5.6, 5.7, 8.0   |
| MRS HBase        | MRS 3.*.*   |
| MRS Hive         | MRS 3.*.*   |
| Oracle           | 11g   |
| PostgreSQL       | 11.0  |
| SQL Server       | 2012, 2014, 2016, 2017  |
| DWS              | 1.3.4   |
| HIVE             | 2.3.2   |
| HANA             | 1.0   |
| MongoDB          | 3.4   |
| Redis            | 2.8.x, 3.x.x  |

## 4.2 Connecting to an API Data Source

### Overview

ROMA Connect can use API as a data source for data integration tasks. Before using the API data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **API** and click **Next**.
4. Configure the data source connection information.

**Table 4-3** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Address                 | Enter the request URL of the API, for example, <b>https://example.com/test</b> .<br><b>NOTE</b><br>If the API to be accessed is created during service integration, you are advised not to use the default subdomain name because it can only be called 1000 times every day. If the number of access times exceeds the limit, data sources will fail to be connected. You are advised to use an independent domain name to access the API. |
| Request Mode            | Select the request method of the API.   |



| Parameter           | Description   |
|---------------------|---|
| Authentication Mode | <p>Select the authentication method of the API.</p> <ul style="list-style-type: none"><li>● <b>None:</b> No authentication is required. Any user can call the API.</li><li>● <b>Basic Auth:</b> The username and password are used for authentication. APIs can be called only after the authentication is successful.<ul style="list-style-type: none"><li>– <b>Username:</b> Enter the username required for basic authentication.</li><li>– <b>Password:</b> Enter the password required for basic authentication.</li></ul></li><li>● <b>AppKey Auth:</b> The AppKey and AppSecret are used to sign the request. The API can be called only after the signature verification is successful.<ul style="list-style-type: none"><li>– <b>App Authentication Type</b><br/><b>Default:</b> The AppKey and AppSecret are used to sign the request. The API can be called only after the signature verification is successful. This mode is used by default for APIs of APIC.<br/><b>Secret:</b> The AppKey and AppSecret are used to authenticate the request.<br/><b>Jwt:</b> The AppKey, AppSecret, and the API calling timestamp are used to generate signature information, and the AppKey, signature and timestamp are used for authentication.</li><li>– <b>AppKey:</b> Enter the AppKey required for AppKey authentication.</li><li>– <b>AppSecret:</b> Enter the AppSecret required for AppKey authentication.</li></ul></li><li>● <b>OAuth 2.0:</b> When an API is called, the request header must contain the authorization information. The API can be called only after the secret in the header is successfully verified.<ul style="list-style-type: none"><li>– <b>Authorization Type:</b> The token information for authorization is obtained through the access token URL. Select an authorization type.</li><li>– <b>Access Token URL:</b> Enter the access token URL of OAuth 2.0 authentication.</li><li>– <b>Client ID:</b> Enter the client ID required for OAuth 2.0 authentication.</li><li>– <b>Client Key:</b> Enter the client key required for OAuth 2.0 authentication.</li><li>– <b>Scope:</b> This parameter is used to define the permission of the application on the API.</li></ul></li><li>● <b>Secret:</b> A password is used for authentication. The request header carries the secret information. The</li></ul> |

| Parameter | Description  |
|-----------|--|
|           | <p>API for obtaining the secret information in the header can be called only after being verified.<br/> <b>Secret:</b> Enter the secret required for secret authentication.</p> <ul style="list-style-type: none"> <li> <b>MD5:</b> The verification information signature is obtained after the body parameters are encrypted. The request body carries the signature information. The API for obtaining the signature information can be called only after being verified.<br/> <b>Secret:</b> Enter the secret required for MD5 authentication.         </li> <li> <b>HMAC:</b> The verification information signature is obtained after the body parameters are encrypted. The request body carries the signature information. The API for obtaining the signature information can be called only after being verified.<br/> <b>Secret:</b> Enter the secret required for HMAC authentication.         </li> <li> <b>Other authentication modes:</b> These authentication modes are customized by business partners, such as <b>Key Top Auth</b>, <b>Hik Vision Auth</b>, and <b>li He Auth</b>. Set the authentication parameters based on the customized authentication mode you select.         </li> </ul> |

The following is an example of connecting to an API released by ROMA Connect. The API authentication mode is **App authentication**.

Set the authentication mode to **AppKey Auth**, and retain **Default** for the APP authentication type. AppKey and AppSecret are the key and secret of the integration application authorized by the API.

**Figure 4-1** API data source configuration example

The screenshot shows a configuration form with the following fields and values:

- Address:**
- Request Mode:**
- Authentication Mode:**
- App Authentication Type:**
- AppKey:**
- AppSecret:**

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.

- If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check whether the address, request mode, authentication mode, AppKey, and AppSecret are correct, and whether the API can be accessed. Then, click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.3 Connecting to an ActiveMQ Data Source

### Overview

ROMA Connect can use ActiveMQ as a data source for data integration tasks. Before using the ActiveMQ data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **ActiveMQ** and click **Next**.
4. Configure the data source connection information.

**Table 4-4** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Broker List             | Enter the broker connection address of ActiveMQ, that is, the IP address and port number of the host. Click <b>Add Address</b> to add addresses. |
| Username                | Enter the username for connecting to ActiveMQ.   |
| Password                | Enter the password for connecting to ActiveMQ.   |

| Parameter             | Description  |
|-----------------------|--|
| SSL Authentication    | Determine whether to use SSL authentication for the connection between ROMA Connect and ActiveMQ.  |
| Authentication Mode   | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Select the SSL authentication mode. <ul style="list-style-type: none"> <li>• <b>One-way SSL</b>: Only the client (ROMA Connect) authenticates the server (ActiveMQ).</li> <li>• <b>Two-way SSL</b>: The client (ROMA Connect) and server (ActiveMQ) authenticate each other.</li> </ul> |
| Trust Store           | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>This parameter specifies the trust store used by the client during SSL authentication. It must match the private key used by the server.  |
| Trust Store Password  | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the password of the trust store.  |
| Keystore              | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>This parameter specifies the keystore used by the client during SSL two-way authentication. It must match the public key used by the server.  |
| Keystore Password     | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>Enter the password of the keystore.   |
| Keystore Key Password | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>Enter the password of the keystore key. If this parameter is not specified, the keystore key password is the same as the keystore password.   |

The following is an example of connecting to an ActiveMQ in one-way SSL authentication mode. You need to upload the trust store.

**Figure 4-2** ActiveMQ data source configuration example

The screenshot shows a configuration form for an ActiveMQ data source. It contains the following elements:

- Broker List:** A text input field containing "10.10.10.1" followed by a slash and another text input field containing "3303". Below it is a button with a plus sign and the text "Add Address".
- Username:** A text input field containing "admin".
- Password:** A text input field with masked characters ".....".
- SSL Authentication:** Two radio buttons: "Disable" (unselected) and "Enable" (selected).
- Authentication Mode:** Two buttons: "One-way SSL" (selected, highlighted in blue) and "Two-way SSL" (unselected).
- Trust Store:** A text input field containing "client.truststore.jks (0.00B)" with a green checkmark icon to its right. To the right of this field is an "Upload" button. Below the field is a green checkmark icon and the text "File uploaded successfully. client.truststore.jks".
- Trust Store Password:** A text input field with the placeholder text "Enter the trust store password."

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.4 Connecting to an ArtemisMQ Data Source

### Overview

ROMA Connect can use ArtemisMQ as a data source for data integration tasks. Before using the ArtemisMQ data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.

2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **ArtemisMQ** and click **Next**.
4. Configure the data source connection information.

**Table 4-5** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Address      | Enter the IP address and port number of the ArtemisMQ.  |
| Username                | Enter the username for connecting to ArtemisMQ.   |
| Password                | Enter the password for connecting to ArtemisMQ.   |
| SSL Authentication      | Determine whether to use SSL authentication for the connection between ROMA Connect and ArtemisMQ.  |
| Authentication Mode     | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Select the SSL authentication mode. <ul style="list-style-type: none"><li>• <b>One-way SSL</b>: Only the client (ROMA Connect) authenticates the server (ArtemisMQ).</li><li>• <b>Two-way SSL</b>: The client (ROMA Connect) and server (ArtemisMQ) authenticate each other.</li></ul> |
| Trust Store             | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>This parameter specifies the trust store used by the client during SSL authentication. It must match the private key used by the server.   |
| Trust Store Password    | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the password of the trust store.   |
| Keystore                | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>This parameter specifies the keystore used by the client during SSL two-way authentication. It must match the public key used by the server.   |

| Parameter             | Description  |
|-----------------------|--|
| Keystore Password     | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>Enter the password of the keystore.   |
| Keystore Key Password | This parameter is mandatory only if <b>SSL Authentication Mode</b> is set to <b>Two-way SSL</b> .<br>Enter the password of the keystore key. If this parameter is not specified, the keystore key password is the same as the keystore password. |

The connection configuration of the ArtemisMQ data source is similar to that of ActiveMQ. For details about the configuration example, see [Figure 4-2](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.5 Connecting to a DB2 Data Source

### Overview

ROMA Connect can use the DB2 database as a data source for data integration tasks. Before using the DB2 data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **DB2** and click **Next**.
4. Configure the data source connection information.

**Table 4-6** Data source connection information

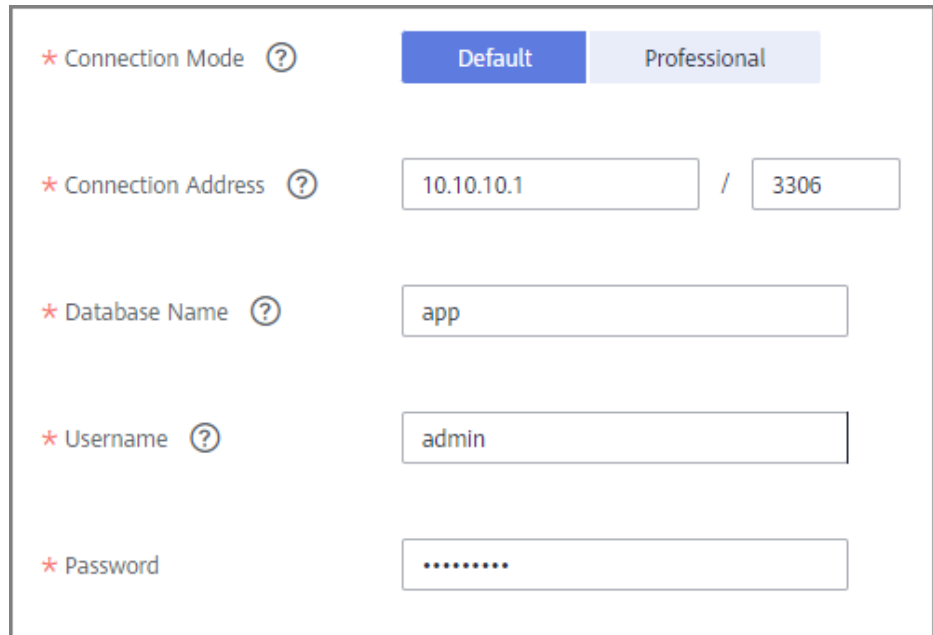
| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Mode         | Select a connection mode for the database. <ul style="list-style-type: none"><li>• <b>Default</b></li><li>• <b>Professional:</b> The JDBC mode is used for database connection.</li></ul>   |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.  |
| Database Name           | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.  |
| Connection String       | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the DB2 database, for example, <code>jdbc:db2://{hostname}:{port}/{dbname}</code> . <ul style="list-style-type: none"><li>• <code>{hostname}</code> indicates the connection address of the database.</li><li>• <code>{port}</code> indicates the port number for connecting to the database.</li><li>• <code>{dbname}</code> indicates the name of the database to be connected.</li></ul> |
| Username                | Enter the username used to connect to the database.   |
| Password                | Enter the password used to connect to the database.   |

The following shows how to configure data source connection.

- Default mode



**Figure 4-3** Configuration example in default mode

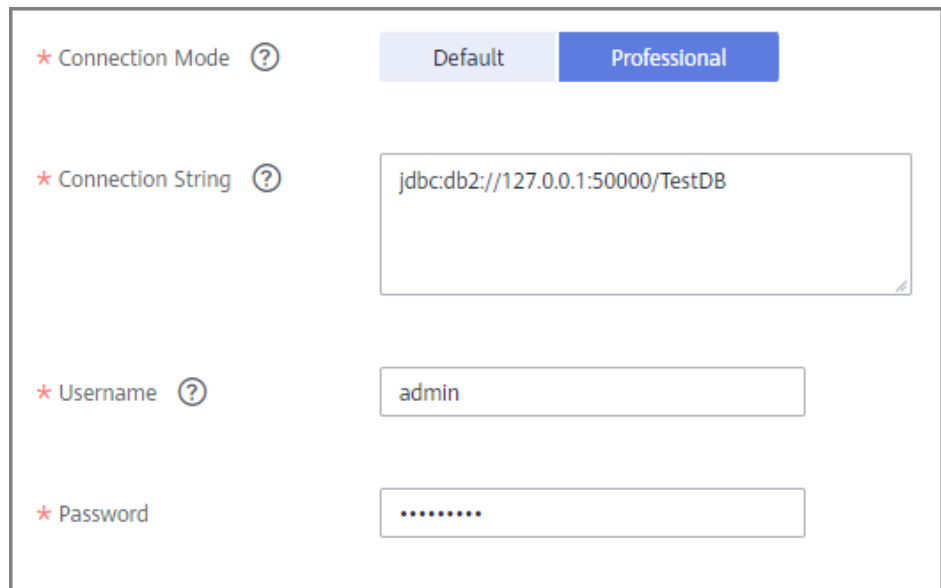


The screenshot shows a configuration form with the following fields and values:

- Connection Mode:** Two tabs, 'Default' (selected) and 'Professional'.
- Connection Address:** A text box containing '10.10.10.1' followed by a slash and a text box containing '3306'.
- Database Name:** A text box containing 'app'.
- Username:** A text box containing 'admin'.
- Password:** A text box with masked characters '.....'.

- Professional mode

**Figure 4-4** Configuration example in professional mode



The screenshot shows a configuration form with the following fields and values:

- Connection Mode:** Two tabs, 'Default' and 'Professional' (selected).
- Connection String:** A text box containing 'jdbc:db2://127.0.0.1:50000/TestDB'.
- Username:** A text box containing 'admin'.
- Password:** A text box with masked characters '.....'.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.6 Connecting to a DIS Data Source

### Overview

ROMA Connect can use the DIS database as a data source for data integration tasks. Before using the DIS data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **DIS** and click **Next**.
4. Configure the data source connection information.

**Table 4-7** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Stream Name             | Enter the name of the DIS stream to be connected.   |
| Data Type               | Select the source data type of the DIS stream. Currently, only JSON is supported.   |
| Configuration Type      | Select the configuration type of the DIS stream. <ul style="list-style-type: none"><li>• <b>Basic</b>: The DIS stream created by the current user in the current region can be called by default.</li><li>• <b>Advanced</b>: DIS streams of different users in different regions can be called.</li></ul> |

| Parameter  | Description   |
|------------|---|
| AK         | Enter the AK of the user to which the DIS stream belongs. The AK and SK are long-term identity credentials of the user on the cloud service platform. For details on how to obtain the AK and SK, see <a href="#">Access Keys</a> .   |
| SK         | Enter the SK of the user to which the DIS stream belongs. The AK and SK are long-term identity credentials of the user on the cloud service platform. For details on how to obtain the AK and SK, see <a href="#">Access Keys</a> .   |
| Endpoint   | This parameter is mandatory only if <b>Configuration Type</b> is set to <b>Advanced</b> .<br>Enter the endpoint of the DIS stream in the format such as https://dis.region1.example.com. For details on how to obtain the endpoint, see <a href="#">Regions and Endpoints</a> . |
| Region     | This parameter is mandatory only if <b>Configuration Type</b> is set to <b>Advanced</b> .<br>Enter the region where the DIS stream is located.  |
| Project ID | This parameter is mandatory only if <b>Configuration Type</b> is set to <b>Advanced</b> .<br>Enter the ID of the project to which the DIS stream belongs. For details about how to obtain the project ID, see <a href="#">API Credentials</a> .                                 |

The following is an example of connecting to a DIS in other regions.

**Figure 4-5** DIS data source configuration example

The screenshot shows a configuration form for a DIS data source. The fields are as follows:

- Stream Name:** doc-test
- Data Type:** JSON
- Configuration Type:** Basic and Advanced (Advanced is selected)
- Endpoint:** https://dis.region1.example.com
- Region:** region1
- Project ID:** 195e56f[redacted]ff98c021d
- AK:** G0WB[redacted]RZKSICX
- SK:** [redacted]

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.7 Connecting to a DWS Data Source

### Overview

ROMA Connect can use the DWS data source for creating data integration tasks or data APIs. Before using the DWS data source, you need to connect it to ROMA Connect.

#### NOTICE

Use the PostgreSQL data source to connect to a DWS database. (The DWS data source is to be unreleased and thus not recommended.)

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **DWS** and click **Next**.
4. Configure the data source connection information.

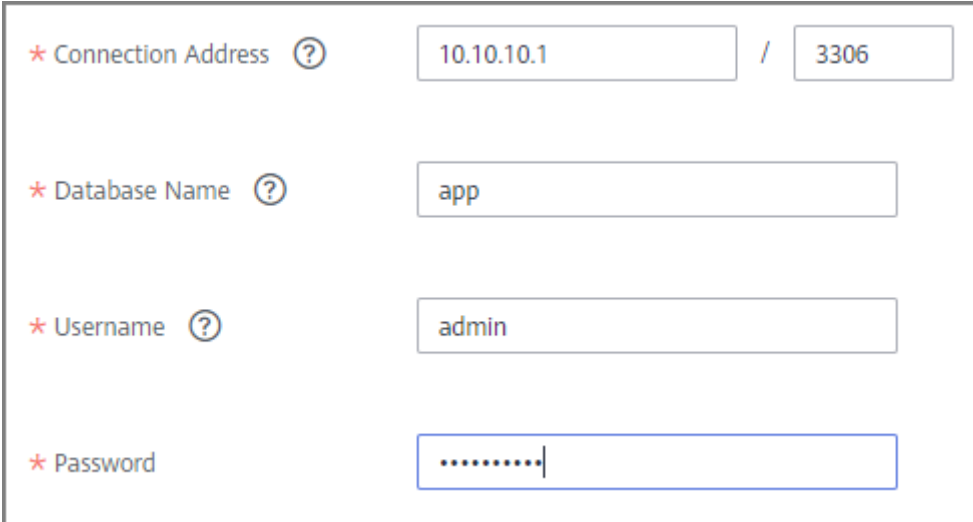
**Table 4-8** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |

| Parameter          | Description  |
|--------------------|--|
| Description        | Enter the descriptive information.   |
| Connection Address | Enter the IP address and port number for connecting the database in the DWS cluster. |
| Database Name      | Enter the name of the database to be connected in the DWS cluster.                   |
| Username           | Enter the username for connecting to the database.                                   |
| Password           | Enter the password for connecting to the database.                                   |

The following figure shows how to configure DWS data source connection.

**Figure 4-6** Example of DWS data source connection configuration



The screenshot shows a configuration form for a DWS data source connection. It contains four rows of input fields, each with a red asterisk and a question mark icon to its left. The first row is for 'Connection Address', with two input boxes containing '10.10.10.1' and '3306' separated by a slash. The second row is for 'Database Name' with one input box containing 'app'. The third row is for 'Username' with one input box containing 'admin'. The fourth row is for 'Password' with one input box containing a series of dots and a cursor.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.8 Connecting to the DM Data Source

### Overview

ROMA Connect can use the DM database as a data source for data integration tasks. Before using the DM data source, you need to connect it to ROMA Connect.

## Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **DM** and click **Next**.
4. Configure the data source connection information.

**Table 4-9** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Mode         | Select a connection mode for the database. <ul style="list-style-type: none"><li>• <b>Default</b></li><li>• <b>Professional</b>: The JDBC mode is used for database connection.</li></ul>  |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.   |
| Connection String       | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the DM database, for example, <i>jdbc:dm://{hostname}:{port}</i> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the connection address of the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li></ul> |
| Username                | Enter the username used to connect to the database.  |
| Password                | Enter the password used to connect to the database.  |

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.9 Connecting to a Gauss100 Data Source

### Overview

ROMA Connect can use the Gauss100 data source for creating data integration tasks or data APIs. Before using the Gauss100 data source, connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **Gauss100** and click **Next**.
4. Configure the data source connection information.

**Table 4-10** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.    |
| Description             | Enter a brief description of the data source.                           |

| Parameter          | Description   |
|--------------------|---|
| Connection Mode    | Select a database connection mode. <ul style="list-style-type: none"><li>• <b>Default:</b> The system automatically concatenates data source connection character strings based on your configured data.</li><li>• <b>Professional:</b> You need to specify the data source connection string manually.</li></ul>   |
| Version Number     | Select the version number of the database to be connected. V100R003C20 and V300R001C00 are supported.   |
| Connection Address | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.  |
| Database Name      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.  |
| Connection String  | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the Gauss100 database, in the format of <i>jdbc:postgresql://host:port/dbname</i> . <ul style="list-style-type: none"><li>• <i>host</i> indicates the IP address for connecting to the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>dbname</i> indicates the name of the Gauss100 database to be connected.</li></ul> |
| Username           | Enter the username for logging in to the database.  |
| Password           | Enter the password for logging in to the database.  |

The connection configuration of the Gauss100 data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.



## 4.10 Connecting to an FTP Data Source

### Overview

ROMA Connect can use FTP as a data source for data integration tasks. Before using the FTP data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **FTP** and click **Next**.
4. Configure the data source connection information.

**Table 4-11** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.                                 |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.                                    |
| Description             | Enter a brief description of the data source.   |
| Protocol                | Select the protocol for connecting to the FTP data source. The options are <b>FTP</b> and <b>SFTP</b> . |
| Connection Address      | Enter the IP address and port number for connecting to the FTP data source.                             |
| Username                | Enter the username used for logging in to the FTP data source.  |
| Password                | Enter the password used for logging in to the FTP data source.  |

| Parameter       | Description  |
|-----------------|--|
| Connection Mode | <p>This parameter is mandatory only if <b>Protocol</b> is set to <b>FTP</b>.</p> <p>Select the connection mode of the FTP data source.</p> <ul style="list-style-type: none"> <li>• <b>Active:</b> The FTP server actively connects to the data port on the FTP client.</li> <li>• <b>Passive:</b> The FTP server waits for the FTP client to connect to its data port.</li> </ul> |

The following figure shows how to configure data source connection.

**Figure 4-7** FTP data source configuration example

The screenshot shows a configuration form for an FTP data source. It contains the following fields:

- \* Protocol:** A dropdown menu with 'FTP' selected.
- \* Connection Address:** Two input fields separated by a slash. The first contains '10.10.10.1' and the second contains '3306'. There is a help icon (?) next to the label.
- \* Username:** An input field containing 'admin'. There is a help icon (?) next to the label.
- \* Password:** An input field with masked characters (dots).
- \* Connection Mode:** A dropdown menu with 'Active' selected.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.11 Connecting to an HL7 Data Source

### Overview

ROMA Connect can use HL7 (a health information exchange standard) as a data source for data integration tasks. Before using the HL7 data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **HL7** and click **Next**.
4. Configure the data source connection information.

**Table 4-12** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Direction               | Select the direction of the HL7 data source used in data integration. The options are <b>Source</b> and <b>Destination</b> .  |
| SSL Authentication      | Determine whether to use SSL authentication for the connection between ROMA Connect and HL7.  |
| Keystore                | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> and <b>Direction</b> is set to <b>Source</b> .<br>This parameter specifies the keystore used by the server (ROMA Connect) during SSL two-way authentication. It must match the public key used by the client (HL7). |

| Parameter             | Description   |
|-----------------------|---|
| Keystore Password     | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> and <b>Direction</b> is set to <b>Source</b> .<br>Enter the password of the keystore.   |
| Keystore Key Password | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> and <b>Direction</b> is set to <b>Source</b> .<br>Enter the password of the keystore key.   |
| Trust Store           | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> and <b>Direction</b> is set to <b>Destination</b> .<br>This parameter specifies the trust store used by the client (ROMA Connect) during SSL authentication. It must match the private key used by the server (HL7).  |
| Trust Store Password  | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> and <b>Direction</b> is set to <b>Destination</b> .<br>Enter the password of the trust store.   |
| Whitelist             | This parameter is mandatory only if <b>Direction</b> is set to <b>Source</b> .<br>Determine whether to allow data to be synchronized to the source HL7 server in the whitelist.   |
| Whitelist Host Server | This parameter is mandatory only if <b>Direction</b> is set to <b>Source</b> and <b>Whitelist</b> is set to <b>Enable</b> .<br>Enter the IP address of the HL7 server that is allowed for data synchronization.   |
| Connection Address    | This parameter is mandatory only if <b>Direction</b> is set to <b>Destination</b> .<br>Enter the IP address and port number for accessing HL7.<br>To obtain the IP address, locate a data integration task whose source is HL7, view the task details, and view <b>Access Address</b> in the source information.<br>To obtain the port, locate a data integration task whose source is HL7, view the task details, and view <b>Access Port</b> in the source information. |

The following figure shows an example of the data source connection configuration when the data source is the destination and SSL authentication is enabled. A trust store file needs to be uploaded.

**Figure 4-8** HL7 data source configuration example

The screenshot shows a configuration form for an HL7 data source. It includes the following fields and options:

- Direction:** Radio buttons for 'Source' and 'Destination', with 'Destination' selected.
- SSL Authentication:** Radio buttons for 'Disable' and 'Enable', with 'Enable' selected.
- Keystore:** A text input field containing 'test.ke (8.00B)' with a green checkmark, and an 'Upload' button. Below it, a message states 'File uploaded successfully. test.ke'.
- Keystore Password:** A password input field with masked characters '.....'.
- Keystore Key Password:** A password input field with masked characters '.....'.
- Connection Address:** Two input fields separated by a slash, containing 'wrggf' and '3306'.

5. Click **Create**.

## 4.12 Connecting to a HANA Data Source

### Overview

ROMA Connect can use the HANA data source for creating data integration tasks or data APIs. Before using the HANA data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **HANA** and click **Next**.
4. Configure the data source connection information.

**Table 4-13** Data source connection information

| Parameter | Description   |
|-----------|---|
| Name      | Enter a data source name. Using naming rules facilitates future search. |

| Parameter               | Description   |
|-------------------------|---|
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Mode         | Select a connection mode for the database. <ul style="list-style-type: none"><li>• <b>Default</b></li><li>• <b>Professional:</b> JDBC is used for database connection.</li></ul>  |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.  |
| Database Name           | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.  |
| Connection String       | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the HANA database, for example, <i>jdbc:sap://{hostname}:{port}?databaseName={dbname}</i> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the connection address of the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>{dbname}</i> indicates the name of the database to be connected.</li></ul> |
| Username                | Enter the username for connecting to the database.  |
| Password                | Enter the password for connecting to the database.  |

The connection configuration of the HANA data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.13 Connecting to a HIVE Data Source

### Overview

ROMA Connect can use HIVE as a data source to create data APIs. Before using the HIVE data source, you need to connect it to ROMA Connect.

#### NOTE

Only a maximum of one million data records can be integrated.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **HIVE** and click **Next**.
4. Configure the data source connection information.

**Table 4-14** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Mode         | Select a connection mode for the database. <ul style="list-style-type: none"><li>• <b>Default</b></li><li>• <b>Professional</b>: JDBC is used for database connection.</li></ul> |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.                                       |

| Parameter         | Description  |
|-------------------|--|
| Database Name     | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.   |
| Connection String | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the HIVE database, for example, <i>jdbc:hive2://{hostname}:{port}/{dbname}</i> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the connection address of the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>{dbname}</i> indicates the name of the database to be connected.</li></ul> |
| Username          | Enter the username used to connect to the database.  |
| Password          | Enter the password used to connect to the database.  |

The connection configuration of the HIVE data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.14 Connecting to an LDAP Data Source

### Overview

ROMA Connect can use LDAP as a data source for data integration tasks. Before using the LDAP data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.



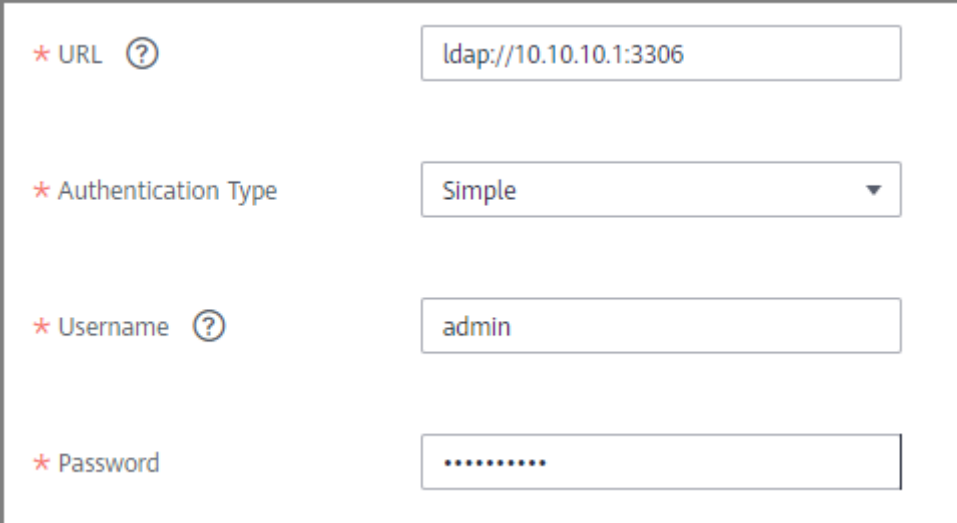
## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **LDAP** and click **Next**.
4. Configure the data source connection information.

**Table 4-15** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| URL                     | Enter the IP address and port number of the server, in the format of <i>ldap://ip:port</i> . <ul style="list-style-type: none"><li>• <i>ip</i> indicates the access address of the database.</li><li>• <i>port</i> indicates the access port number of the database.</li></ul> |
| Authentication Type     | Select the security authentication mode of the database. Only <b>Simple</b> is available.  |
| Username                | Enter the username used for logging in to the database.  |
| Password                | Enter the password used for logging in to the database.  |

The following figure shows how to configure data source connection.

**Figure 4-9** LDAP data source configuration exampleA screenshot of a web form for configuring an LDAP data source. The form contains four rows of input fields, each with a red asterisk and a question mark icon to its left. The first row is labeled 'URL' and contains the text 'ldap://10.10.10.1:3306'. The second row is labeled 'Authentication Type' and contains a dropdown menu with 'Simple' selected. The third row is labeled 'Username' and contains the text 'admin'. The fourth row is labeled 'Password' and contains a series of dots representing a masked password.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.15 Connecting to an IBM MQ Data Source

### Overview

ROMA Connect can use IBM MQ as a data source for data integration tasks. Before using the IBM MQ data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **IBM MQ** and click **Next**.
4. Configure the data source connection information.

**Table 4-16** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Address      | Enter the IP address and port number for connecting the IBM MQ.  |
| Queue Manager           | Enter the name of the IBM MQ manager.  |
| CCSID                   | Enter the CCSID of the IBM MQ manager.   |
| Stream Name             | Enter the stream name for connecting to the IBM MQ manager.  |
| Username                | Enter the username for connecting to IBM MQ.   |
| Password                | Enter the password for connecting to IBM MQ.   |
| SSL Authentication      | Determine whether to use SSL authentication for the connection between ROMA Connect and IBM MQ.  |
| Cipher Suite            | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the name of the cipher suite used in the queue manager stream specified by <b>Stream Name</b> .   |
| Trust Store             | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the trust store used by the client (ROMA Connect) during SSL authentication. It must match the private key used by the server (IBM MQ). |
| Trust Store Password    | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the password of the trust store.  |

The following figure shows an example of data source connection configuration when SSL is enabled. A trust store file needs to be uploaded.

**Figure 4-10** Example of IBM MQ data source connection configuration

The screenshot shows a configuration form for an IBM MQ data source. It includes the following fields and options:

- Connection Address**: 10.10.10.1 / 3306
- Queue Manager**: MQ\_ZAVIER
- CCSID**: 819
- Stream Name**: CLNTCONN
- Username**: admin
- Password**: [Redacted]
- SSL Authentication**:  Disable  Enable
- Cipher Suite**: CLNTCONNtest
- Trust Store**: doc.jks (0.00B) [Green checkmark] [Upload button]. Below it, a message reads: "File uploaded successfully. doc.jks".
- Trust Store Password**: [Redacted]

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.16 Connecting to a Kafka Data Source

### Overview

ROMA Connect can use the Kafka data source for data integration tasks. Before using the Kafka data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **Kafka** and click **Next**.
4. Configure the data source connection information.

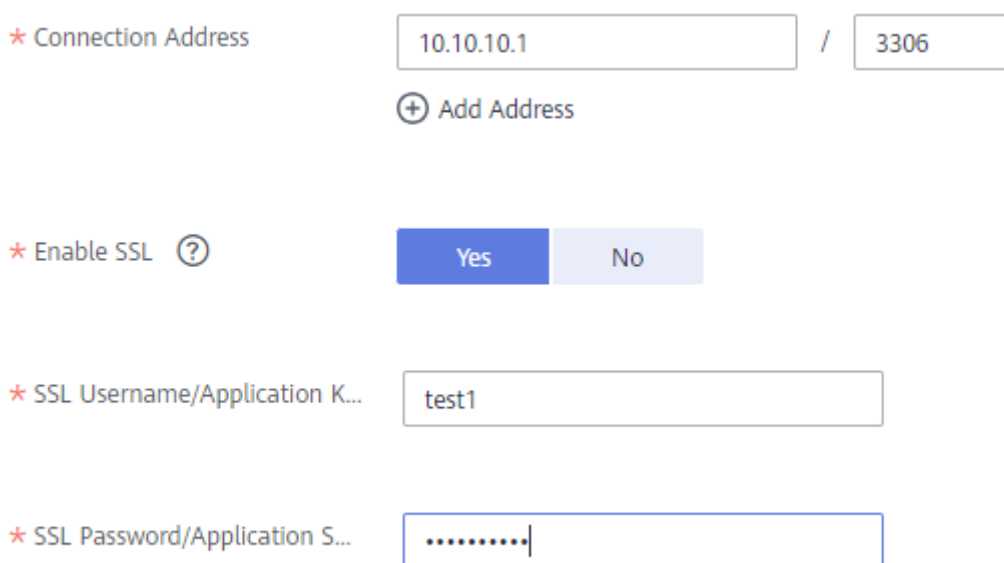
**Table 4-17** Data source connection information

| Parameter                    | Description  |
|------------------------------|--|
| Name                         | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format              | Default: utf-8   |
| Integration Application      | Select the integration application to which the data source belongs.   |
| Description                  | Enter a brief description of the data source.  |
| Connection Address           | Enter the IP address and port number for connecting Kafka.<br>If Kafka has multiple brokers, click <b>Add Address</b> to enter the connection addresses.   |
| Enable SASL_SSL              | Whether to use SASL_SSL authentication for the connection between ROMA Connect and Kafka.  |
| Username/<br>Application Key | Mandatory for <b>Enable SASL_SSL</b> set to <b>Yes</b> .<br>Enter the username used for SASL_SSL authentication. If ROMA Connect MQS is used as the Kafka data source, the username is the key of the integration application. |

| Parameter                          | Description   |
|------------------------------------|---|
| Password/<br>Application<br>Secret | Mandatory for <b>Enable SASL_SSL</b> set to <b>Yes</b> .<br>Enter the password used for SASL_SSL authentication. If ROMA Connect MQS is used as the Kafka data source, the username is the secret of the integration application. |

The following figure shows an example of data source connection configuration when SASL\_SSL is enabled.

**Figure 4-11** Kafka data source configuration example



The screenshot shows a configuration form with the following elements:

- Connection Address:** A field containing '10.10.10.1' followed by a slash and a field containing '3306'. Below it is a button with a plus sign and the text 'Add Address'.
- Enable SSL:** A toggle switch with a question mark icon. The 'Yes' option is selected and highlighted in blue.
- SSL Username/Application K...:** A text input field containing 'test1'.
- SSL Password/Application S...:** A password input field with masked characters '.....' and a cursor.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.17 Connecting to a MySQL Data Source

### Overview

ROMA Connect can use the MySQL database as a data source for data integration tasks or data API creation. Before using the MySQL data source, you need to connect it to ROMA Connect.

## Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MySQL** and click **Next**.
4. Configure the data source connection information.

**Table 4-18** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Mode         | Select a database connection mode. <ul style="list-style-type: none"><li>• <b>Default:</b> The system automatically concatenates data source connection character strings based on your configured data.</li><li>• <b>Professional:</b> You need to specify the data source connection string manually.<ul style="list-style-type: none"><li>- If the time zone of data written to a service is different from the default time zone of the MySQL database, you need to use the JDBC mode to connect to the database and specify the time zone in the JDBC connection string. For example, jdbc:mysql://10.10.10.1:3306/testfdi?serverTimezone=GMT%2B8.</li><li>- If the encoding character set of data written by a service is different from the default encoding character set in the MySQL database, you need to use the JDBC mode to connect to the database and specify the encoding character set in the JDBC connection string. For example, jdbc:mysql://10.10.10.1:3306/testfdi?characterEncoding=utf8.</li></ul></li></ul> |

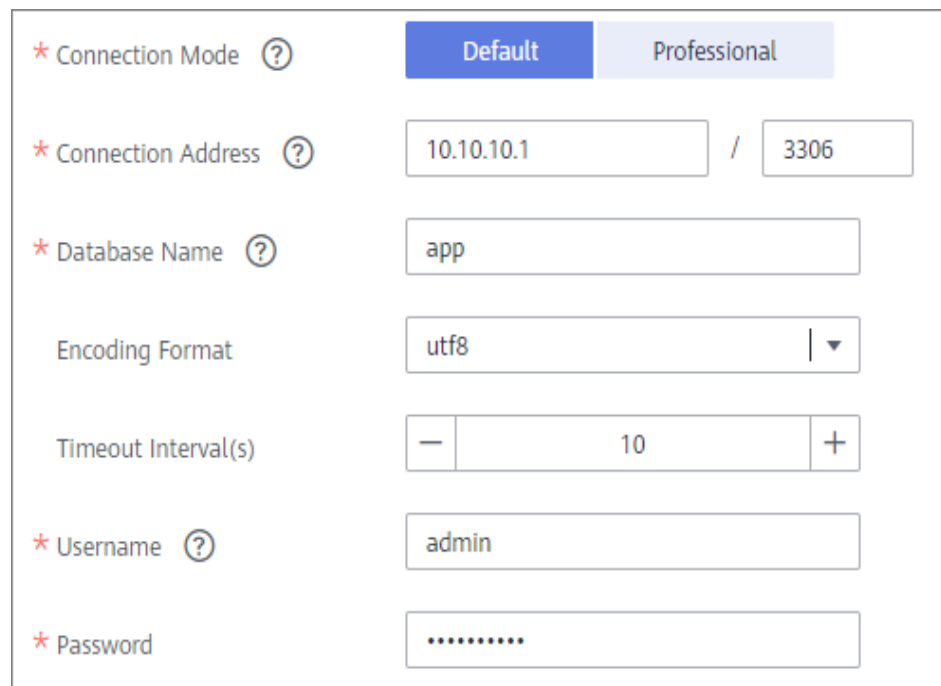
| Parameter          | Description  |
|--------------------|--|
| Connection Address | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.   |
| Database Name      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.   |
| Encoding Format    | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the encoding format used by the database.   |
| Timeout Interval   | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the timeout interval for connecting to the database, in seconds.  |
| Connection String  | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the MySQL database, for example, <i>jdbc:mysql://{hostname}:{port}/{dbname}</i> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the IP address for connecting to the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>{dbname}</i> indicates the name of the database to be connected.</li></ul> |
| Username           | Enter the username used to connect to the database.  |
| Password           | Enter the password used to connect to the database.  |

The following shows how to configure data source connection.

- Default mode



**Figure 4-12** Configuration example in default mode



\* Connection Mode ? Default Professional

\* Connection Address ?  /

\* Database Name ?

Encoding Format  ▾

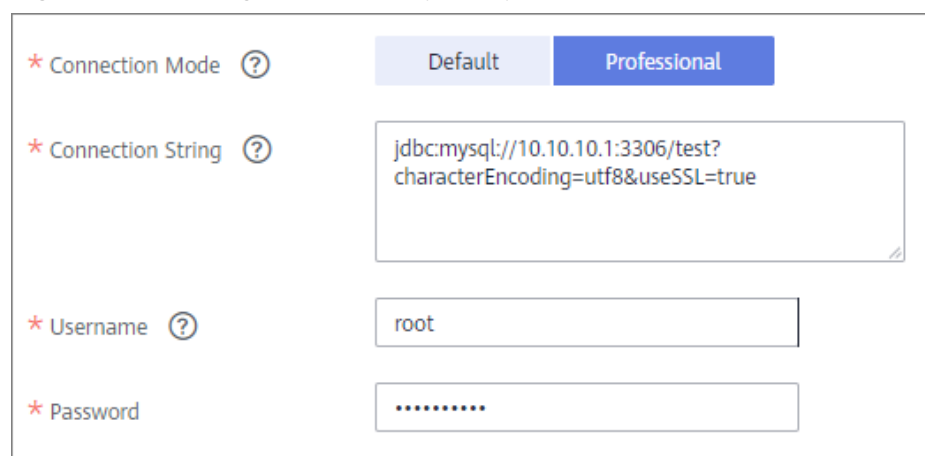
Timeout Interval(s)  - +

\* Username ?

\* Password

- Professional mode

**Figure 4-13** Configuration example in professional mode



\* Connection Mode ? Default Professional

\* Connection String ?

\* Username ?

\* Password

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.18 Connecting to a MongoDB Data Source

### Overview

ROMA Connect can use the MongoDB database as a data source for data integration tasks or data API creation. Before using the MongoDB data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MongoDB** and click **Next**.
4. Configure the data source connection information.

**Table 4-19** Data source connection information

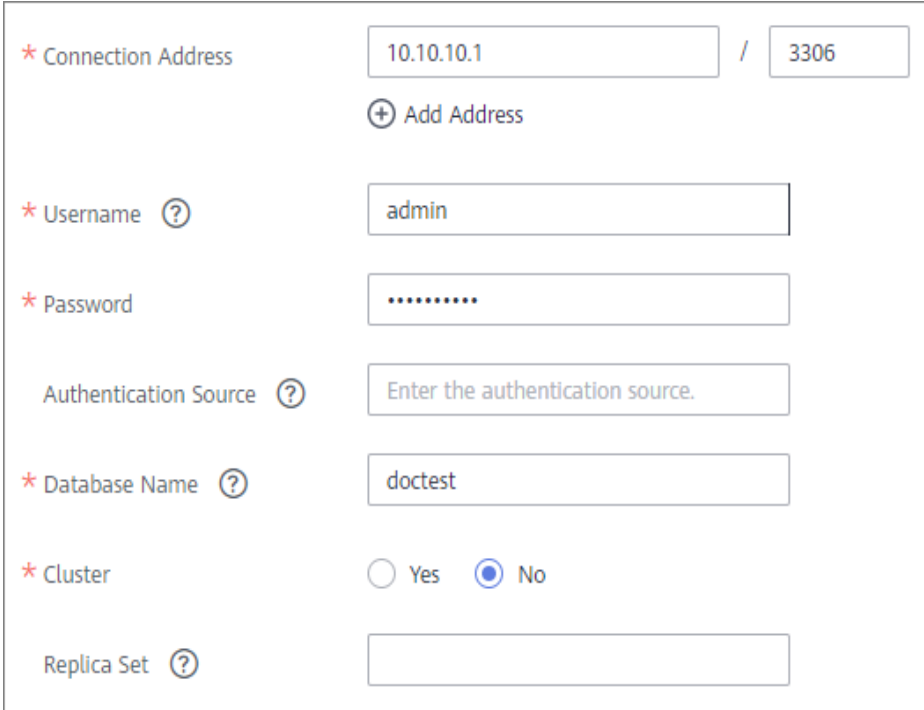
| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Address      | Enter the IP address and port number of MongoDB, in the format of IP:PORT.<br>If MongoDB has multiple replica sets, click <b>Add Address</b> to enter the connection addresses.  |
| Username                | Enter the username used to connect to the database.  |
| Password                | Enter the password used to connect to the database.  |
| Authentication Source   | Enter the name of the MongoDB database used for access authentication, that is, the MongoDB database used to authenticate access users. If this parameter is not specified, the database to be connected is used to authenticate users by default. |

| Parameter     | Description   |
|---------------|---|
| Database Name | Enter the name of the database to be connected.   |
| Cluster       | Determine whether the MongoDB database to be connected uses the cluster mode. <ul style="list-style-type: none"><li>If the MongoDB database uses the cluster mode, select <b>Yes</b> for <b>Cluster</b>.</li><li>If the MongoDB database uses the replica set or single-node mode, select <b>No</b> for <b>Cluster</b>.</li></ul> |
| Replica Set   | This parameter is mandatory only if <b>Cluster</b> is set to <b>No</b> . If the MongoDB database uses the replica set mode, enter the replica set name.   |

The following is an example of connecting to a MongoDB data source in single-node mode.

The authentication source is **admin**. That is, the admin database of MongoDB is used to authenticate user root. After the authentication is successful, the user can connect to the doctest database of MongoDB.

**Figure 4-14** MongoDB data source configuration example



The screenshot shows a configuration form for a MongoDB data source. The fields are as follows:

- Connection Address:** 10.10.10.1 / 3306. There is an "Add Address" button below.
- Username:** admin
- Password:** masked with dots
- Authentication Source:** Enter the authentication source.
- Database Name:** doctest
- Cluster:** Radio buttons for Yes and No, with No selected.
- Replica Set:** Empty text input field.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.

- If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.19 Connecting to an MQS Data Source

### Overview

ROMA Connect can use the MQS data source for data integration tasks. Before using the MQS data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MQS** and click **Next**.
4. Configure the data source connection information.

**Table 4-20** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Address      | Select the MQS private connection address in the current instance.   |
| Enable SSL              | Determine whether to use SSL authentication for the connection between ROMA Connect and MQS.<br>Set this parameter to <b>Yes</b> if <b>SSL</b> is enabled and <b>Intra-VPC Plaintext Access</b> is disabled. In other cases, set this parameter to <b>No</b> . |

| Parameter                           | Description  |
|-------------------------------------|--|
| SSL Username/<br>Application Key    | This parameter is mandatory only if <b>Enable SSL</b> is set to <b>Yes</b> .<br>Enter the username used for SSL authentication. If ROMA Connect MQS is used as the MQS data source, the username is the key of the integration application.    |
| SSL Password/<br>Application Secret | This parameter is mandatory only if <b>Enable SSL</b> is set to <b>Yes</b> .<br>Enter the password used for SSL authentication. If ROMA Connect MQS is used as the MQS data source, the username is the secret of the integration application. |

The following figure shows an example of data source connection configuration when SSL is enabled.

**Figure 4-15** Example of MQS data source connection configuration

The screenshot shows a configuration form with the following elements:

- Connection Address:** A dropdown menu showing the IP address and port: 192.168.0.123:3033.
- Enable SSL:** A radio button group with 'Yes' selected and 'No' unselected. A help icon (?) is next to the label.
- SSL Username/Application Key:** A text input field containing the value 'test'.
- SSL Password/Application Secret:** A text input field where the password is masked with eight dots.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.20 Connecting to an MRS Hive Data Source

### Overview

ROMA Connect can use the MRS Hive data source for data integration tasks. Before using the MRS Hive data source, you need to connect it to ROMA Connect.

#### NOTE

- If two data integration tasks use MRS data sources of different versions (including MRS Hive, MRS HDFS, and MRS HBase) and Kerberos authentication is enabled for the MRS data sources, the two data integration tasks cannot be executed at the same time. Otherwise, the integration tasks fail.
- Only a maximum of one million data records can be integrated.

### Prerequisites

- Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.
- Kerberos authentication has been enabled for the MRS cluster where the MRS Hive data source is located. The execution permission has been configured for machine-machine interaction users. For details, see [Preparing a Development User](#).

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MRS Hive** and click **Next**.
4. Configure the data source connection information.

**Table 4-21** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.    |
| Description             | Enter a brief description of the data source.                           |

| Parameter                | Description   |
|--------------------------|---|
| HDFS URL                 | Enter the name of the MRS Hive file system to access. <ul style="list-style-type: none"><li>• If the root directory is used, set this parameter to <b>hdfs:///</b>. This operation requires administrator permissions.</li><li>• If the default directory is used, set this parameter to <b>hdfs:///hacluster</b>. This operation requires administrator permissions.</li><li>• If a planned directory is used, set this parameter to the planned directory.</li><li>• If a user database directory is used, for example, <b>/user/hive/testdb</b>, the user must have the permission on the directory.</li></ul> |
| Machine-machine Username | Enter the machine-machine username for connecting to MRS Hive.  |
| Configuration File       | Click <b>Upload</b> to upload the MRS Hive configuration files. For details about how to obtain the files, see "Obtaining MRS Hive configuration files".  |

### Obtaining MRS Hive configuration files

- a. Obtain the **krb5.conf** and **user.keytab** files.

Download the user authentication file from MRS Manager by following the procedure described in [Downloading a User Authentication File](#), and decompress the file to obtain the **krb5.conf** and **user.keytab** files.

- b. Obtain the **hiveclient.properties**, **core-site.xml**, **hdfs-site.xml**, and **hosts** files.

Download the client configuration file from the MRS console by following the procedure described in [Updating a Client Configuration File](#). After the file is decompressed:

- Obtain the **hosts** file from `xxx_Services_ClientConfig_ConfigFiles`.
- Obtain the **hiveclient.properties** file from `xxx_Services_ClientConfig_ConfigFiles > Hive > config`.
- Obtain the **core-site.xml** and **hdfs-site.xml** files from `xxx_Services_ClientConfig_ConfigFiles > HDFS > config`.

Check whether the value of

**dfs.client.failover.proxy.provider.hacluster** in the **hdfs-site.xml** file is

**org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**. If no, change it to this value.

- c. Create a **Version** file.

Create a text file named **Version** without an extension, and add **version=MRS 3.1.0** to the file.

- d. Obtain the MRS Hive configuration files.

Save the obtained files to a new directory and compress them into a **.zip** package. All files are stored in the root directory of the **.zip** package.

 **NOTE**

- The file name contains a maximum of 255 characters, including only letters and digits.
  - The file size cannot exceed 2 MB.
5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
    - If the test result is **Data source connected successfully**, go to the next step.
    - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
  6. Click **Create**.

## 4.21 Connecting to an MRS HDFS Data Source

### Overview

ROMA Connect can use the MRS HDFS data source for data integration tasks. Before using the MRS HDFS data source, you need to connect it to ROMA Connect.

 **NOTE**

If two data integration tasks use MRS data sources of different versions (including MRS Hive, MRS HDFS, and MRS HBase) and Kerberos authentication is enabled for the MRS data sources, the two data integration tasks cannot be executed at the same time. Otherwise, the integration tasks fail.

### Prerequisites

- Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.
- Kerberos authentication has been enabled for the MRS cluster where the MRS HDFS data source is located. The execution permission has been configured for machine-machine interaction users. For details, see [Preparing a Development User](#)

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MRS HDFS** and click **Next**.
4. Configure the data source connection information.



**Table 4-22** Data source connection information

| Parameter                | Description   |
|--------------------------|---|
| Name                     | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format          | Default: utf-8  |
| Integration Application  | Select the integration application to which the data source belongs.  |
| Description              | Enter a brief description of the data source.   |
| HDFS URL                 | Enter the name of the MRS HDFS file system to access. <ul style="list-style-type: none"><li>• If the root directory is used, set this parameter to <b>hdfs:///</b>. This operation requires administrator permissions.</li><li>• If the default directory is used, set this parameter to <b>hdfs:///hacluster</b>. This operation requires administrator permissions.</li><li>• If a planned directory is used, set this parameter to the planned directory.</li><li>• If a user database directory is used, for example, <b>/user/hdfs/testdb</b>, the user must have the permission on the directory.</li></ul> |
| Machine-machine Username | Enter the machine-machine username for connecting to MRS HDFS.  |
| Configuration File       | Click <b>Upload</b> to upload the MRS HDFS configuration file. For details about how to obtain the files, see "Obtaining MRS HDFS configuration files".   |

### Obtaining MRS HDFS configuration files

- a. Obtain the **krb5.conf** and **user.keytab** files.

Download the user authentication file from MRS Manager by following the procedure described in [Downloading a User Authentication File](#), and decompress the file to obtain the **krb5.conf** and **user.keytab** files.
- b. Obtain the **core-site.xml**, **hdfs-site.xml**, and **hosts** files.

Download the client configuration file from the MRS console by following the procedure described in [Updating a Client Configuration File](#). After the file is decompressed:

  - Obtain the **hosts** file from `xxx_Services_ClientConfig_ConfigFiles`.
  - Obtain the **core-site.xml** and **hdfs-site.xml** files from `xxx_Services_ClientConfig_ConfigFiles > HDFS > config`.

Check whether the value of **dfs.client.failover.proxy.provider.hacluster** in the **hdfs-site.xml** file is

**org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider**. If no, change it to this value.

- c. Obtain the MRS HDFS configuration files.

Save the obtained files to a new directory and compress them into a **.zip** package. All files are stored in the root directory of the **.zip** package.

 **NOTE**

- The file name contains a maximum of 255 characters, including only letters and digits.
  - The file size cannot exceed 2 MB.
5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
    - If the test result is **Data source connected successfully**, go to the next step.
    - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
  6. Click **Create**.

## 4.22 Connecting to an MRS HBase Data Source

### Overview

ROMA Connect can use the MRS HBase data source for creating data integration tasks. Before using the MRS HBase data source, you need to connect it to ROMA Connect.

 **NOTE**

If two data integration tasks use MRS data sources of different versions (including MRS Hive, MRS HDFS, and MRS HBase) and Kerberos authentication is enabled for the MRS data sources, the two data integration tasks cannot be executed at the same time. Otherwise, the integration tasks fail.

### Prerequisites

- Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.
- Kerberos authentication has been enabled for the MRS cluster where the MRS HBase data source is located. The execution permission has been configured for machine-machine interaction users. For details, see [Preparing a Development User](#).

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.

3. On the **Default** tab page, select **MRS HBase** and click **Next**.
4. Configure the data source connection information.

**Table 4-23** Data source connection information

| Parameter                | Description  |
|--------------------------|--|
| Name                     | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format          | Default: utf-8   |
| Integration Application  | Select the integration application to which the data source belongs.   |
| Description              | Enter a brief description of the data source.  |
| Machine-machine Username | Enter the machine-machine username for connecting to MRS HBase.  |
| Configuration File       | Click <b>Upload</b> to upload the MRS HBase configuration files. For details about how to obtain the files, see "Obtaining MRS HBase configuration files". |

### Obtaining MRS HBase configuration files

- a. Obtain the **krb5.conf** and **user.keytab** files.  
Download the user authentication file from MRS Manager by following the procedure described in [Downloading a User Authentication File](#), and decompress the file to obtain the **krb5.conf** and **user.keytab** files.
- b. Obtain the **core-site.xml**, **hbase-site.xml**, **hdfs-site.xml**, and **hosts** files.  
Download the client configuration file from the MRS console by following the procedure described in [Updating a Client Configuration File](#). After the file is decompressed:
  - Obtain the **hosts** file from `xxx_Services_ClientConfig_ConfigFiles`.
  - Obtain the **hbase-site.xml** file from `xxx_Services_ClientConfig_ConfigFiles > HBase > config`.
  - Obtain the **core-site.xml** and **hdfs-site.xml** files from `xxx_Services_ClientConfig_ConfigFiles > HDFS > config`.
- c. Generate the MRS HBase configuration file.  
Save the obtained files to a new directory and compress them into a **.zip** package. All files are stored in the root directory of the **.zip** package.

#### NOTE

- The file name contains a maximum of 255 characters, including only letters and digits.
- The file size cannot exceed 2 MB.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.23 Connecting to an MRS Kafka Data Source

### Overview

ROMA Connect can use the MRS Kafka data source for data integration tasks. Before using the MRS Kafka data source, you need to connect it to ROMA Connect.

### Prerequisites

- Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.
- Kerberos authentication has been enabled for the MRS cluster where the MRS Kafka data source is located. The execution permission has been configured for machine-machine interaction users. For details, see [Preparing a Development User](#)

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **MRS Kafka** and click **Next**.
4. Configure the data source connection information.

**Table 4-24** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.    |
| Description             | Enter a brief description of the data source.                           |

| Parameter                | Description   |
|--------------------------|---|
| Machine-machine Username | Enter the machine-machine username for connecting to MRS Kafka.   |
| Configuration File       | Click <b>Upload</b> to upload the MRS Kafka configuration file. For details about how to obtain the files, see "Obtaining MRS Kafka configuration files". |

### Obtaining MRS Kafka configuration files

- a. Obtain the **krb5.conf** and **user.keytab** files.  
Download the user authentication file from MRS Manager by following the procedure described in [Downloading a User Authentication File](#), and decompress the file to obtain the **krb5.conf** and **user.keytab** files.
- b. Obtain the **producer.properties**, **consumer.properties**, and **hosts** files.  
Download the client configuration file from the MRS console by following the procedure described in [Updating a Client Configuration File](#). After the file is decompressed:
  - Obtain the **hosts** file from `xxx_Services_ClientConfig_ConfigFiles`.
  - Obtain the **producer.properties** and **consumer.properties** files from `xxx_Services_ClientConfig_ConfigFiles > Kafka > config`.  
Change **producer.properties** to **fikafka-producer.properties** and **consumer.properties** to **fikafka-consumer.properties**, and copy **bootstrap.servers** in **fikafka-producer.properties** to **fikafka-consumer.properties**.
- c. Obtain the MRS Kafka configuration files.  
Save the obtained files to a new directory and compress them into a **.zip** package. All files are stored in the root directory of the **.zip** package.

#### NOTE

- The file name contains a maximum of 255 characters, including only letters and digits.
- The file size cannot exceed 2 MB.

5. Click **Create**.

## 4.24 Connecting to an OBS Data Source

### Overview

ROMA Connect can use the OBS database as a data source for data integration tasks. Before using the OBS data source, you need to connect it to ROMA Connect.

## Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

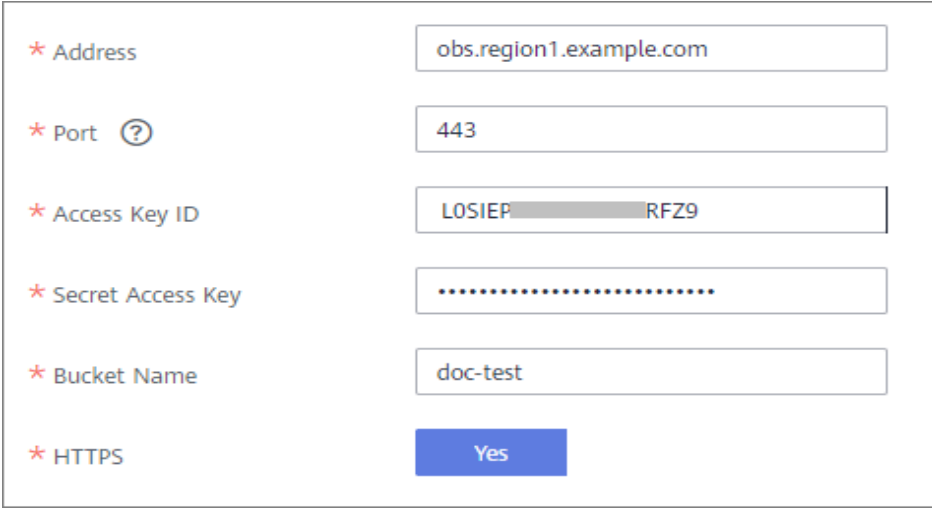
## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **OBS** and click **Next**.
4. Configure the data source connection information.

**Table 4-25** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. It is recommended that you enter a name based on naming rules to facilitate search.           |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter the descriptive information.  |
| Address                 | Enter the endpoint address of the OBS bucket to be connected. You can obtain the endpoint address from the OBS console. |
| Port                    | It has a fixed value of <b>443</b> .  |
| Access Key ID           | Enter the AK of the OBS bucket owner. For details about how to obtain the AK, see <a href="#">My Credentials</a> .      |
| Secret Access Key       | Enter the SK of the OBS bucket owner. For details about how to obtain the SK, see <a href="#">My Credentials</a> .      |
| Bucket Name             | Enter the name of the OBS bucket to be connected.   |
| HTTPS                   | This parameter has a fixed value of <b>Yes</b> , indicating that HTTPS is used.   |

The following is an example of connecting to an OBS data source.

**Figure 4-16** OBS data source configuration example

The screenshot shows a configuration form for an OBS data source. It includes the following fields and options:

- Address:** obs.region1.example.com
- Port:** 443
- Access Key ID:** LOSIEP [REDACTED] RFZ9
- Secret Access Key:** [REDACTED]
- Bucket Name:** doc-test
- HTTPS:** Yes (selected)

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.25 Connecting to an Oracle Data Source

### Overview

ROMA Connect can use the Oracle database as a data source for data integration tasks or data API creation. Before using the Oracle data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **Oracle** and click **Next**.
4. Configure the data source connection information.

**Table 4-26** Data source connection information

| Parameter                 | Description  |
|---------------------------|--|
| Name                      | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format           | Default: utf-8   |
| Integration Application   | Select the integration application to which the data source belongs.   |
| Description               | Enter a brief description of the data source.  |
| Real-Time Synchronization | Determine whether to enable real-time synchronization for the database. Set this parameter to <b>Enable</b> if Oracle is used as the source data source of a composite integration task.   |
| CDC Mode                  | This parameter is mandatory only if <b>Real-Time Synchronization</b> is set to <b>Enable</b> .<br>Select the CDC mode for the database. <ul style="list-style-type: none"> <li>• <b>XStream</b>: The CDC function of the Oracle database in XStream mode is used.</li> <li>• <b>LogMiner</b>: The CDC function of the Oracle database in LogMiner mode is used.</li> </ul> |
| Outbound Server           | This parameter is mandatory only if <b>CDC Mode</b> is set to <b>XStream</b> .<br>Enter the name of the outbound server, which must be the same as that set in <a href="#">Configuring Oracle CDC (XStream)</a> .  |
| Database Mode             | This parameter is mandatory only if <b>CDC Mode</b> is set to <b>LogMiner</b> .<br>Enter the name of the schemas where the data table to be synchronized is located. If this parameter is not specified the database username is used by default.  |
| Pluggable Database        | This parameter is mandatory only if <b>Real-Time Synchronization</b> is set to <b>Enable</b> .<br>Enter the name of the PDB where the data table to be synchronized is located.  |



| Parameter             | Description  |
|-----------------------|--|
| Connection Mode       | <p>Select a database connection mode.</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> The system automatically concatenates data source connection character strings based on your configured data.</li> <li>• <b>Multi-address:</b> You can enter the IP addresses and port numbers of multiple databases. The system automatically concatenates data source connection character strings based on your configured data.</li> <li>• <b>Professional:</b> You need to specify the data source connection string manually.</li> </ul>  |
| Connection Address    | <p>Mandatory only when <b>Connection Mode</b> is set to <b>Default</b>.</p> <p>Enter the IP address and port number for connecting the database.</p>   |
| Database Name         | <p>Mandatory only when <b>Connection Mode</b> is set to <b>Default</b>.</p> <p>Enter the name of the database to be connected.</p>   |
| Database Address List | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Multi-address</b>.</p> <p>Enter the connection address and port number of the distributed database system. Click <b>Add Address</b> to add multiple addresses and port numbers.</p>   |
| Service Name          | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Multi-address</b>.</p> <p>Enter the service name of the distributed database system.</p>  |
| Connection String     | <p>Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b>.</p> <p>Enter a JDBC connection string of the Oracle database, for example,<br/> <code>jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST={hostname})(PORT={port})))(CONNECT_DATA=(SERVER=DEDICATED)(SERVICE_NAME={servicename})))</code>.</p> <ul style="list-style-type: none"> <li>• <i>{hostname}</i> indicates the IP address for connecting to the database.</li> <li>• <i>{port}</i> indicates the port number for connecting to the database.</li> <li>• <i>{servicename}</i> indicates the service name of the Oracle database to be connected.</li> </ul> |
| Username              | Enter the username used to connect to the database.  |
| Password              | Enter the password used to connect to the database.  |

For a common data integration task, the connection configuration of the Oracle data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

For a composite task, the following is an example of connecting to an Oracle data source which is not a CDB database and the CDC mode is **LogMiner**.

**Figure 4-17** Oracle data source configuration example

The screenshot displays the configuration interface for an Oracle data source. It includes the following settings:

- Real-Time Synchronization:** A toggle switch set to **Enable**.
- CDC Mode:** A dropdown menu set to **LogMiner**.
- Database Mode:** A text input field containing **schenaroma**.
- Pluggable Database:** A text input field with the placeholder text **Enter a pluggable database.**
- Connection Mode:** A set of three tabs: **Default** (selected), **Multi-address**, and **Professional**.
- Connection Address:** Two text input fields containing **10.10.10.1** and **1443**, separated by a slash.
- Database Name:** A text input field containing **doctest**.
- Username:** A text input field containing **admin**.
- Password:** A text input field with masked characters (dots).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.26 Connecting to a PostgreSQL Data Source

### Overview

ROMA Connect can use the PostgreSQL/openGauss data source for creating data integration tasks or data APIs. Before using this data source, connect it to ROMA Connect.

## Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **PostgreSQL** and click **Next**.
4. Configure the data source connection information.

**Table 4-27** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Mode         | Select a database connection mode. <ul style="list-style-type: none"><li>• <b>Default:</b> The system automatically concatenates data source connection character strings based on your configured data.</li><li>• <b>Professional:</b> You need to specify the data source connection string manually.</li></ul> |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.  |
| Database Name           | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.  |

| Parameter         | Description   |
|-------------------|---|
| Connection String | <p>Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b>.</p> <p>Enter the JDBC connection string of the PostgreSQL database, for example, <code>jdbc:postgresql://{hostname}:{port}/{dbname}</code>.</p> <ul style="list-style-type: none"><li>• <code>{hostname}</code> indicates the IP address for connecting to the database.</li><li>• <code>{port}</code> indicates the port number for connecting to the database.</li><li>• <code>{dbname}</code> indicates the name of the database to be connected.</li></ul> |
| Username          | Enter the username for connecting to the database.  |
| Password          | Enter the password for connecting to the database.  |

The connection configuration of the PostgreSQL data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.27 Connecting to a Redis Data Source

### Overview

ROMA Connect can use the Redis database as a data source for data integration tasks or data API creation. Before using the Redis data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

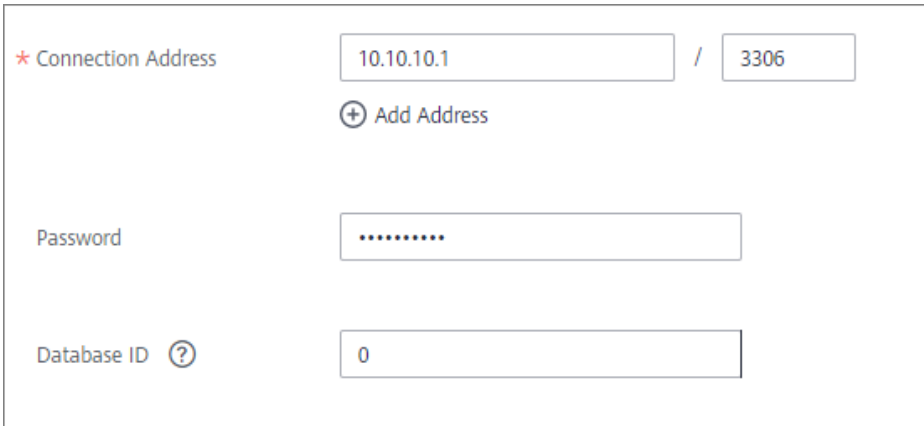
1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.

2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **Redis** and click **Next**.
4. Configure the data source connection information.

**Table 4-28** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Address      | Enter the IP address and port number for connecting Redis. If there are multiple connection addresses, click <b>Add Address</b> to add them.<br><b>NOTE</b><br>If the Redis data source is a cluster Redis, you need to write the addresses of all nodes. If only one node address is written, ROMA Connect regards the Redis as a single-node Redis. As a result, the cluster Redis fails to be accessed. |
| Password                | Enter the password for connecting to Redis.  |
| Database ID             | Enter the ID of the Redis database. If this parameter is not specified, the database numbered <b>0</b> is connected by default.  |

The following figure shows how to configure Redis data source connection.

**Figure 4-18** Redis data source configuration example

The screenshot shows a configuration form for a Redis data source. It contains the following fields and elements:

- Connection Address:** A field containing "10.10.10.1" followed by a slash and a field containing "3306". Below this is a blue button with a plus sign and the text "Add Address".
- Password:** A text input field with the content masked by seven dots.
- Database ID:** A text input field containing the number "0". To the left of the field is a question mark icon.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.

- If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.28 Connecting to a RabbitMQ Data Source

### Overview

ROMA Connect can use RabbitMQ as a data source for data integration tasks. Before using the RabbitMQ data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **RabbitMQ** and click **Next**.
4. Configure the data source connection information.

**Table 4-29** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.    |
| Description             | Enter a brief description of the data source.                           |
| Connection Address      | Enter the IP address and port number of the RabbitMQ.                   |
| Username                | Enter the username for connecting to RabbitMQ.                          |
| Password                | Enter the password for connecting to RabbitMQ.                          |
| Virtual Host            | Enter the name of the RabbitMQ virtual host to be connected.            |

| Parameter               | Description   |
|-------------------------|---|
| SSL Authentication      | Determine whether to use SSL authentication for the connection between ROMA Connect and RabbitMQ.   |
| Authentication Protocol | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Select the protocol used for SSL authentication. This parameter has a fixed value of <b>TLS</b> .  |
| Trust Store             | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Select the trust store used by the client (ROMA Connect) during SSL authentication. It must match the private key used by the server (RabbitMQ). |
| Trust Store Password    | This parameter is mandatory only if <b>SSL Authentication</b> is set to <b>Enable</b> .<br>Enter the password of the trust store.   |

The connection configuration of the RabbitMQ data source is similar to that of the ActiveMQ data source. For details, see [Connecting to an ActiveMQ Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.29 Connecting to a RocketMQ Data Source

### Overview

ROMA Connect can use the RocketMQ data source for data integration tasks. Before using the RocketMQ data source, connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

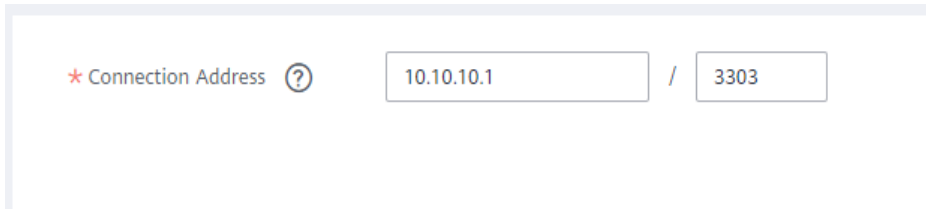
1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.

2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **RocketMQ** and click **Next**.
4. Configure the data source connection information.

**Table 4-30** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.    |
| Description             | Enter the descriptive information.                                      |
| Connection Address      | Enter the IP address and port number for connecting RocketMQ.           |

The following figure shows how to configure data source connection.

**Figure 4-19** Configuration example for RocketMQ data source

The screenshot shows a configuration form for a RocketMQ data source. The 'Connection Address' field is highlighted with a red asterisk and a question mark icon. The input field contains the IP address '10.10.10.1' followed by a slash and the port number '3303'.

**NOTE**

The RocketMQ data source does not support access through port mapping.

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.



## 4.30 Connecting to an SAP Data Source

### Overview

ROMA Connect supports the SAP data source to connect with SAP systems through RFC for data integration tasks. Before using the SAP data source, connect it to ROMA Connect. SAP-related permissions are required for data integration tasks to access an SAP data source. For details, see [Table 4-31](#).

**Table 4-31** SAP permissions

| Object Class Text                       | Authorization Object | Authorization Object Text               | Authorization Field | Text                    | Value  |
|---|----------------------|---|---------------------|-------------------------|--|
| Cross-application authorization objects | S_RFC                | Cross-application authorization objects | RFC_TY<br>PE        | Allowed RFC object type | FUNC   |
| Cross-application authorization objects | S_RFC                | Cross-application authorization objects | RFC_N<br>AME        | Allowed RFC object name | <ul style="list-style-type: none"> <li>• DDIF_FIELDINFO_GET</li> <li>• RFCPING</li> <li>• RFC_GET_FUNCTION_INTERFACE</li> <li>• /SAPDS/RFC_READ_TABLE2</li> <li>• SAPTUNE_GET_SUMMARY_STATISTIC</li> <li>• TH_WPINFO</li> <li>• RFC_FUNCTION_SEARCH_WITHGROUP</li> </ul> |
| Cross-application authorization objects | S_RFC                | Cross-application authorization objects | ACTVT               | Activity                | 16   |

| Object Class Text                       | Authorization Object | Authorization Object Text                                 | Authorization Field | Text                       | Value |
|---|----------------------|---|---------------------|----------------------------|-------|
| Cross-application authorization objects | S_TC<br>ODE          | Checking the transaction code when the transaction starts | TCD                 | Transaction code           | SM50  |
| Basic: management                       | S_TA<br>BU_N<br>AM   | Generic table access                                      | ACTVT               | Activity                   | 3     |
| Basic: management                       | S_TA<br>BU_N<br>AM   | Generic table access                                      | Table               | Table name                 | *     |
| Basic: management                       | S_AD<br>MI_F<br>CD   | System authorization                                      | S_ADM<br>I_FCD      | System management function | STOR  |

## Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **SAP** and click **Next**.
4. Configure the data source connection information.

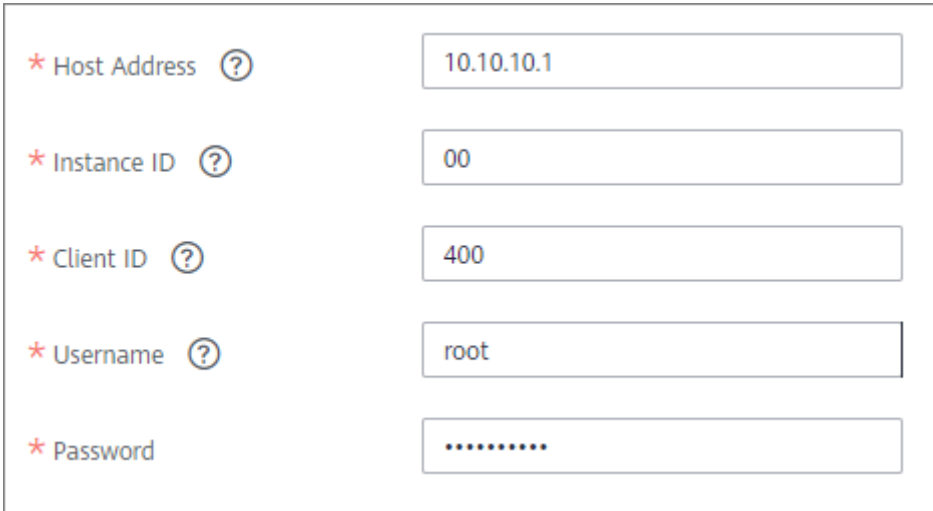
**Table 4-32** Data source connection information

| Parameter       | Description   |
|-----------------|---|
| Name            | Enter a data source name. Using naming rules facilitates future search. |
| Encoding Format | Default: utf-8  |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| IP Address              | Enter the IP address of the SAP application server.   |
| Instance ID             | Enter the instance number of the SAP application server. The instance number consists of two letters or digits. |
| Customer Code           | Enter the ID of the client to be connected in the SAP system. The client ID consists of three digits.           |
| Username                | Enter the username for connecting to the SAP client.  |
| Password                | Enter the password for connecting to the SAP client.  |

The following figure shows how to configure SAP data source connection.

**Figure 4-20** SAP data source configuration example



The screenshot shows a configuration form for an SAP data source. It contains five rows, each with a label, a red asterisk, a question mark icon, and a text input field. The values entered in the fields are: Host Address (10.10.10.1), Instance ID (00), Client ID (400), Username (root), and Password (represented by eight dots).

|                  |            |
|------------------|------------|
| * Host Address ? | 10.10.10.1 |
| * Instance ID ?  | 00         |
| * Client ID ?    | 400        |
| * Username ?     | root       |
| * Password       | .....      |

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.31 Connecting to an SNMP Data Source

### Overview

ROMA Connect can use SNMP as a data source for data integration tasks. Before using the SNMP data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **SNMP** and click **Next**.
4. Configure the data source connection information.

**Table 4-33** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Connection Address      | Enter the IP address and port number of the SNMP.   |
| Network Protocol        | Select the network protocol used by SNMP. The options are <b>UDP</b> and <b>TCP</b> .   |
| Version                 | Select the SNMP version. <ul style="list-style-type: none"><li>• <b>0</b>: SNMPv1</li><li>• <b>1</b>: SNMPv2</li><li>• <b>3</b>: SNMPv3</li></ul> |

| Parameter               | Description  |
|-------------------------|--|
| Community               | This parameter is mandatory only if <b>Version</b> is set to <b>0</b> or <b>1</b> .<br>Enter the SNMP community name, which is used for identity authentication when you access the SNMP management agent. The community name is equivalent to the access password.  |
| Security Username       | This parameter is mandatory only if <b>Version</b> is set to <b>3</b> .<br>Enter the security name used for connecting to the SNMP.  |
| Context Name            | This parameter is mandatory only if <b>Version</b> is set to <b>3</b> .<br>Enter the name of the context to be accessed in SNMP.   |
| Security Level          | This parameter is mandatory only if <b>Version</b> is set to <b>3</b> .<br>Select the security level of SNMP. <ul style="list-style-type: none"><li>• <b>1</b>: Neither authentication nor encryption is required.</li><li>• <b>2</b>: Authentication is required but encryption is not required.</li><li>• <b>3</b>: Both authentication and encryption are required.</li></ul> |
| Authentication Protocol | This parameter is mandatory only if <b>Security Level</b> is set to <b>2</b> or <b>3</b> .<br>Select the authentication protocol for connecting to SNMP. The options are <b>MD5</b> and <b>SHA1</b> .  |
| Authentication Key      | This parameter is mandatory only if <b>Security Level</b> is set to <b>2</b> or <b>3</b> .<br>Enter the authentication key for connecting to SNMP.   |
| Encryption Protocol     | This parameter is mandatory only if <b>Security Level</b> is set to <b>3</b> .<br>Select the encryption protocol for connecting to SNMP. The options are <b>DES</b> and <b>TRIDES</b> .  |
| Encryption Key          | This parameter is mandatory only if <b>Security Level</b> is set to <b>3</b> .<br>Enter the encryption key for connecting to SNMP.   |

The following is an example for connecting to SNMPv3.

**Figure 4-21** SNMP data source configuration example

|                                     |  |
|-------------------------------------|--|
| * Connection Address <span>?</span> | <input type="text" value="10.10.10.1"/> / <input type="text" value="161"/> |
| * Network Protocol                  | <input type="text" value="UDP"/>   |
| * Version                           | <input type="text" value="3"/>   |
| * Security Name                     | <input type="text" value="doctest"/>                                       |
| * Context Name                      | <input type="text" value="testcontext"/>                                   |
| * Security Level                    | <input type="text" value="3"/>   |
| * Authentication Protocol           | <input type="text" value="MD5"/>   |
| * Authentication Key                | <input type="text" value="....."/>   |
| * Encryption Protocol               | <input type="text" value="DES"/>   |
| * Encryption Key                    | <input type="text" value="....."/>   |

5. Click **Create**.

## 4.32 Connecting to a SQL Server Data Source

### Overview

ROMA Connect can use the SQL Server database as a data source for data integration tasks or data API creation. Before using the SQL Server data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.

3. On the **Default** tab page, select **SQL Server** and click **Next**.
4. Configure the data source connection information.

**Table 4-34** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Mode         | Select a database connection mode. <ul style="list-style-type: none"><li>• <b>Default:</b> The system automatically concatenates data source connection character strings based on your configured data.</li><li>• <b>Professional:</b> You need to specify the data source connection string manually.</li></ul>  |
| Connection Address      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.   |
| Database Name           | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.   |
| Connection String       | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the SQL Server database, for example, <i>jdbc:sqlserver://{hostname}:{port};DatabaseName={dbname}</i> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the IP address for connecting to the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>{dbname}</i> indicates the name of the database to be connected.</li></ul> |
| Username                | Enter the username used to connect to the database.  |
| Password                | Enter the password used to connect to the database.  |

The connection configuration of the SQL Server data source is similar to that of the DB2 data source. For details, see [Connecting to a DB2 Data Source](#).

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.33 Connecting to a GaussDB(for MySQL) Data Source

### Overview

ROMA Connect can use the GaussDB(for MySQL) database as a data source for data integration tasks. Connect to the GaussDB(for MySQL) data source before using it.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or **create one** first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **GaussDB(for MySQL)** and click **Next**.
4. Configure the data source connection information.

**Table 4-35** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a data source name. Using naming rules facilitates future search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter a brief description of the data source.  |
| Connection Mode         | Select a connection mode for the database. <ul style="list-style-type: none"><li>• <b>Default</b></li><li>• <b>Professional</b>: JDBC is used for database connection.</li></ul> |



| Parameter          | Description   |
|--------------------|---|
| Connection Address | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the IP address and port number for connecting the database.  |
| Connection String  | Mandatory only when <b>Connection Mode</b> is set to <b>Professional</b> .<br>Enter the JDBC connection string of the GaussDB(for MySQL) database, for example, <b><code>jdbc:mysql://{hostname}:{port}/{dbname}?characterEncoding=utf8&amp;useSSL=true</code></b> . <ul style="list-style-type: none"><li>• <i>{hostname}</i> indicates the IP address for connecting to the database.</li><li>• <i>{port}</i> indicates the port number for connecting to the database.</li><li>• <i>{dbname}</i> indicates the name of the database to be connected.</li></ul> |
| Database Name      | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the name of the database to be connected.  |
| Encoding Format    | Mandatory only when <b>Connection Mode</b> is set to <b>Default</b> .<br>Enter the encoding format used by the database.  |
| Username           | Enter the username for connecting to the database.  |
| Password           | Enter the password for connecting to the database.  |

The following shows how to configure data source connection.

- Default mode

**Figure 4-22** Configuration example in default mode

|                        |   |
|------------------------|---|
| * Connection Mode ?    | <input checked="" type="radio"/> Default <input type="radio"/> Professional |
| * Connection Address ? | <input type="text" value="10.10.10.1"/> / <input type="text" value="3306"/> |
| * Database Name ?      | <input type="text" value="app"/>  |
| Encoding Format        | <input type="text" value="utf8"/>   |
| * Username ?           | <input type="text" value="root"/>   |
| * Password             | <input type="password" value="....."/>                                      |

- Professional mode

**Figure 4-23** Configuration example in professional mode

|                       |   |
|-----------------------|---|
| * Connection Mode ?   | <input type="radio"/> Default <input checked="" type="radio"/> Professional                           |
| * Connection String ? | <input type="text" value="jdbc:mysql://10.10.10.1:3306/test?characterEncoding=utf8&amp;useSSL=true"/> |
| * Username ?          | <input type="text" value="root"/>   |
| * Password            | <input type="password" value="....."/>  |

5. Click **Check Connectivity** to check the connectivity between ROMA Connect and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
6. Click **Create**.

## 4.34 Connecting to a WebSocket Data Source

### Overview

ROMA Connect can use WebSocket as a data source for data integration tasks. Before using the WebSocket data source, you need to connect it to ROMA Connect.

### Prerequisites

Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Default** tab page, select **WebSocket** and click **Next**.
4. Configure the data source connection information.

**Table 4-36** Data source connection information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a data source name. Using naming rules facilitates future search.   |
| Encoding Format         | Default: utf-8  |
| Integration Application | Select the integration application to which the data source belongs.  |
| Description             | Enter a brief description of the data source.   |
| Address                 | Enter the IP address of WebSocket. <ul style="list-style-type: none"><li>• ws (WebSocket) indicates a plaintext connection. If a plaintext connection is used, the URL must be in the <i>ws://example.com</i> or <i>IP:PORT</i> format.</li><li>• wss (WebSocket Secure) indicates an encrypted connection. If an encrypted connection is used, the URL must be in the <i>wss://example.com</i> or <i>IP:PORT</i> format.</li></ul> |

| Parameter           | Description  |
|---------------------|--|
| Authentication Mode | Select the authentication mode used for connecting to WebSocket. <ul style="list-style-type: none"> <li>• <b>None:</b> No authentication is required. Anyone can access WebSocket.</li> <li>• <b>Basic Auth:</b> A user must enter the username and password for authentication. After the authentication is successful, the user is allowed to access WebSocket.</li> </ul> |
| Username            | This parameter is mandatory only if <b>Authentication Mode</b> is set to <b>Basic Auth</b> .<br>Enter the username for connecting to WebSocket.  |
| Password            | This parameter is mandatory only if <b>Authentication Mode</b> is set to <b>Basic Auth</b> .<br>Enter the password for connecting to WebSocket.  |

The following figure uses the authentication mode **Basic Auth** as an example to describe how to configure data source access.

**Figure 4-24** WebSocket data source configuration example

The screenshot shows a configuration form with the following fields:

- Address:** ws://example.com
- Authentication Mode:** Basic Auth
- Username:** test
- Password:** Masked with dots

- Click **Check Connectivity** to check the connectivity between FDI and the data source.
  - If the test result is **Data source connected successfully**, go to the next step.
  - If the test result is **Failed to connect to the data source**, check the data source status and connection parameters, and click **Recheck** until the connection is successful.
- Click **Create**.

## 4.35 Connecting to a Custom Data Source

### Overview

ROMA Connect can use a custom connector as a data source for data integration tasks. Before using a custom data source, you need to connect it to ROMA Connect.

### Prerequisites

- Each connected data source must belong to an integration application. Ensure that an integration application is available before connecting a data source, or [create one](#) first.
- A connector instance is available. Otherwise, [create a connector](#) and [publish a connector](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Data Sources**. In the upper right corner of the page, click **Access Data Source**.
3. On the **Custom** tab page, select the connector to be used as the custom data source and click **Next**.
4. Configure the data source connection information.

**Table 4-37** Data source connection information

| Parameter               | Description  |
|-------------------------|--|
| Data Source Name        | Enter a data source name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Encoding Format         | Default: utf-8   |
| Integration Application | Select the integration application to which the data source belongs.   |
| Description             | Enter the descriptive information.   |
| Connector Instance      | Select a published connector instance under the connector.   |
| Other parameters        | Set the subsequent configuration items based on the data source information defined when you create the connector.<br><br>You can find the connector on the <b>Assets</b> page and view the data source metadata of the connector. |

The following is an example of connecting to a custom data source for reading email data.

The **email server**, **protocol**, **user**, and **password** are the data source parameters defined in the connector.

**Figure 4-25** Custom data source configuration example

|                      |                |
|----------------------|----------------|
| * Connector Instance | connector_mail |
| * email server       | 10.10.10.10    |
| * protocol           | smtp           |
| * user               | docuser        |
| * password           | *****          |

5. Click **Create**.

# 5 Data Integration Guide

---

[Usage Introduction](#)

[Connecting to Data Sources](#)

[Creating a Common Data Integration Task](#)

[Creating a Composite Data Integration Task](#)

[Creating a Flow Data Integration Task](#)

[Starting or Stopping a Data Integration Task](#)

[Managing a Data Integration Task](#)

[Connectors](#)

## 5.1 Usage Introduction

### Function Description

FDI is a data integration component of ROMA Connect and supports data integration and conversion between multiple data sources. ROMA Connect has the following advantages for data integration:

- **Access of multiple data source types**

By default, ROMA Connect supports connections of multiple types of data sources, such as relational databases, big data storage, semi-structured storage, and message systems. For details about the supported data source types, see [Data Sources Supported by Data Integration Task](#).

If the default data source types provided by ROMA Connect cannot meet your data integration requirements, you can customize data sources. For details on how to customize a data source, see [Connecting to a Custom Data Source](#).

- **Flexible integration modes**

ROMA Connect supports the following integration modes:

- **Scheduled:** ROMA Connect periodically obtains data from the source based on a task schedule and then integrates the data to the destination.

- Real-time: ROMA Connect integrates data generated at the source to the destination in real time.

For details about the data source types supported in the two integration modes, see [Connecting to Data Sources](#).

- **Custom mapping rules**

When converting data fields from the source to destination, you can customize mapping rules. For example, you can replicate one data column in source data to multiple data columns and then integrate these columns to the destination.

- **Data integration between different network environments**

FDI allows data sources at the source and destination to come from different network environments that are not interconnected. For example, if the data source at the source comes from an on-premises data center and the data source at the destination comes from a VPC on the cloud, FDI can access both the data sources, implementing data integration between different network environments.

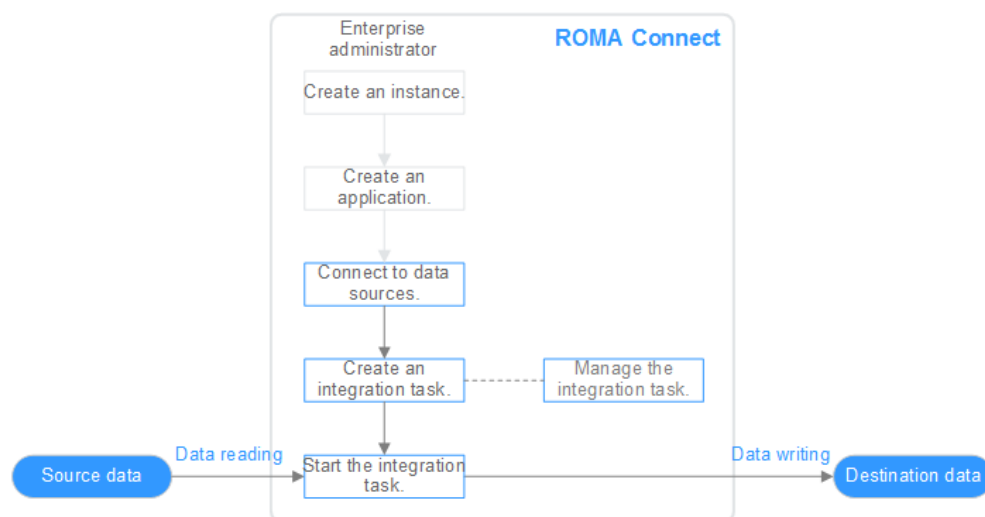
- **Resumable transmission of real-time tasks**

After a fault on the source or destination is rectified or a task is manually restarted, FDI automatically resumes data collection from the last interrupted position to prevent data loss.

## Process Flow

The following figure shows how data integration is performed using ROMA Connect.

**Figure 5-1** Using ROMA Connect for data integration



1. You have **created a ROMA Connect instance and an integration application**.
2. **Access data sources.**

Access data sources at the source and destination to ensure that data can be read from the source and written to the destination.



### 3. Create an integration task.

A data integration task defines detailed rules for data integration from the source to destination. The rules include data source types on both the source and destination, mapping rules of data fields, and filtering conditions for data integration. ROMA Connect allows you to create the following data integration tasks:

- **Common Data Integration Task:** Supports all default data source types and two integration modes: scheduled and real-time. For database data sources, only one data table at the source can be integrated to one data table at the destination each time.
- **Composite Data Integration Task:** Uses the Change Data Capture (CDC) to implement real-time and incremental synchronization of data from the source to the destination. Multiple data tables at the source can be integrated to multiple data tables at the destination. Currently, the following relational databases are supported: Oracle, MySQL, and SQL Server. For details, see [CDC Configurations](#).

### 4. Start the integration task.

- After a scheduled task is started, ROMA Connect integrates data on a scheduled basis. During the first execution, all source data that meets the conditions is integrated to the destination. Then, full data that meets the conditions or only incremental data will be integrated based on the task configuration.
- After a real-time task is started, ROMA Connect continuously detects data changes at the source. During the first execution, all source data that meets the conditions is integrated to the destination. Subsequently, only new data will be integrated to the destination each time.

## 5.2 Connecting to Data Sources

### Overview

Before creating a data integration task, you need to connect data sources at the source and destination to ROMA Connect so that data can be read from the source and written to the destination.

### Prerequisites

- Before accessing a data source, ensure that the ROMA Connect instance can connect to the network where your data source resides.
  - Same VPC: The instance can directly access the data source.
  - Two VPCs in the same region: Connect the instance and the data source using a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Connect the instance and the data source using a peering connection. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.
- To enable cross-VPC communication over the private network, [configure the routes](#) between the instance and the subnet where the data source resides.

## Connecting to a Data Source

The configuration for data source connection varies depending on data source types. After a data source is connected, click the data source name to view the data source details, which include relevant task information.

 **NOTE**

- ROMA Connect data integration is applicable to data type conversion and integration between heterogeneous data sources. **DRS** is recommended for data migration and synchronization between mainstream databases. In scenarios where data such as relational database data, big data, and text is migrated to the data lake, **CDM** is recommended.

**Table 5-1** Configurations for data source connections

| Data Source Connection                                 | Common Task Integration Mode (Source) | Common Task Integration Mode (Destination) | Composite Task Integration Mode (Source) | Composite Task Integration Mode (Destination) |
|--|---------------------------------------|--|--|---|
| <a href="#">Connecting to an API Data Source</a>       | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to an ActiveMQ Data Source</a>  | Real-Time                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to an ArtemisMQ Data Source</a> | Real-Time                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a DB2 Data Source</a>        | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a DIS Data Source</a>        | Real-Time                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a DWS Data Source</a>        | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to the DM Data Source</a>       | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a Gauss100 Data Source</a>   | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to an FTP Data Source</a>       | Scheduled                             | Scheduled                                  | -  | -   |
| <a href="#">Connecting to an HL7 Data Source</a>       | Real-Time                             | Scheduled, Real-Time                       | -  | -   |

| <b>Data Source Connection</b>                          | <b>Common Task Integration Mode (Source)</b> | <b>Common Task Integration Mode (Destination)</b> | <b>Composite Task Integration Mode (Source)</b> | <b>Composite Task Integration Mode (Destination)</b> |
|--|--|---|---|--|
| <a href="#">Connecting to a HANA Data Source</a>       | Scheduled                                    | Scheduled, Real-Time                              | Scheduled                                       | Scheduled  |
| <a href="#">Connecting to an LDAP Data Source</a>      | Scheduled                                    | -   | -   | -  |
| <a href="#">Connecting to an IBM MQ Data Source</a>    | Real-Time                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to a Kafka Data Source</a>      | Real-Time                                    | Scheduled, Real-Time                              | -   | Real-Time  |
| <a href="#">Connecting to a MySQL Data Source</a>      | Scheduled                                    | Scheduled, Real-Time                              | Scheduled, Real-Time                            | Scheduled, Real-Time                                 |
| <a href="#">Connecting to a MongoDB Data Source</a>    | Scheduled                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an MQS Data Source</a>       | Real-Time                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an MRS Hive Data Source</a>  | Scheduled                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an MRS HDFS Data Source</a>  | Scheduled                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an MRS HBase Data Source</a> | Scheduled                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an MRS Kafka Data Source</a> | Real-Time                                    | Scheduled, Real-Time                              | -   | -  |
| <a href="#">Connecting to an OBS Data Source</a>       | Scheduled                                    | Scheduled   | -   | -  |
| <a href="#">Connecting to an Oracle Data Source</a>    | Scheduled                                    | Scheduled, Real-Time                              | Scheduled, Real-Time                            | Scheduled, Real-Time                                 |
| <a href="#">Connecting to a PostgreSQL Data Source</a> | Scheduled                                    | Scheduled, Real-Time                              | Scheduled, Real-Time                            | Scheduled, Real-Time                                 |

| Data Source Connection   | Common Task Integration Mode (Source) | Common Task Integration Mode (Destination) | Composite Task Integration Mode (Source) | Composite Task Integration Mode (Destination) |
|--|---------------------------------------|--|--|---|
| <a href="#">Connecting to a Redis Data Source</a>              | -                                     | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a RabbitMQ Data Source</a>           | Real-Time                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a RocketMQ Data Source</a>           | Real-Time                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to an SAP Data Source</a>               | Scheduled                             | -  | -  | -   |
| <a href="#">Connecting to an SNMP Data Source</a>              | Scheduled                             | -  | -  | -   |
| <a href="#">Connecting to a SQL Server Data Source</a>         | Scheduled                             | Scheduled, Real-Time                       | Scheduled, Real-Time                     | Scheduled, Real-Time                          |
| <a href="#">Connecting to a GaussDB(for MySQL) Data Source</a> | Scheduled                             | Scheduled, Real-Time                       | -  | -   |
| <a href="#">Connecting to a WebSocket Data Source</a>          | Real-Time                             | -  | -  | -   |
| <a href="#">Connecting to a Custom Data Source</a>             | Scheduled                             | Scheduled                                  | -  | -   |

## 5.3 Creating a Common Data Integration Task

### 5.3.1 Configuring Basic Information

#### Overview

You can convert data among different data sources by creating a data integration task on ROMA Connect. Based on your configured data integration task, ROMA Connect determines how to integrate data in a specific data table at the source to a data table at the destination.

A data integration task includes basic information, schedule (optional), source information, destination information, mapping information, fault information storage (optional), and post-integration operation (optional). This topic describes how to configure basic information and create a schedule for a data integration task.

## Prerequisites

- The source and destination data sources have been connected to ROMA Connect. (For details, see [Connecting to Data Sources](#).)
- ROMA Connect has the permission to write data to the destination.
- If you need to configure data storage for abnormal synchronization is abnormal, ensure that the OBS data source has been connected. For details, see [Connecting to an OBS Data Source](#).

## Configuring Basic Information

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**. On the displayed page, click **Create Common Task**.
3. On the **Create Task** page, configure basic task information.

**Table 5-2** Basic task information

| Parameter          | Description   |
|--------------------|---|
| Task Name          | Enter a task name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Description        | Enter a brief description of the task.  |
| Integration Mode   | Select <b>Scheduled</b> as the mode used for data integration. <ul style="list-style-type: none"><li>• <b>Scheduled</b>: A data integration task is executed according to the schedule to integrate data from the source to the destination.</li><li>• <b>Real-Time</b>: The data integration task continuously detects updates to the data at the source and integrates updates to the destination in real time.</li></ul> The data integration mode varies depending on the data source. For details, see <a href="#">Table 5-1</a> . |
| Tag                | Add or select an existing tag to classify tasks for quick search. New tags are saved when you save the task and can be searched directly when you create another task.  |
| Enterprise Project | Select an enterprise project. You can associate an instance with the enterprise project. This configuration option is available only for enterprise accounts.   |

4. Proceed to the next step.

- If **Integration Mode** is set to **Scheduled**, [create a schedule](#).
- If **Integration Mode** is set to **Real-Time**, [configure source information](#).

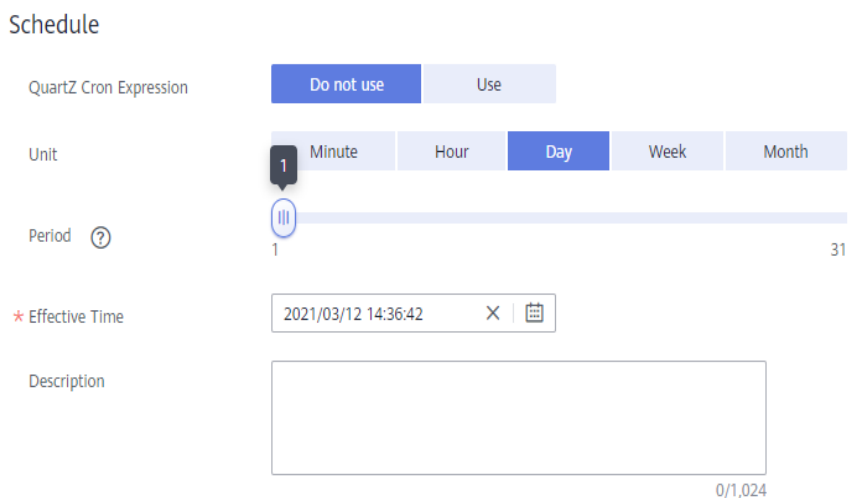
## (Optional) Creating a Schedule

If **Integration Mode** of a task is set to **Scheduled**, you need to configure schedule information for the task.

### NOTE

- If you need to modify a schedule after a data integration task is created, stop the task first.
  - If a large amount of data needs to be synchronized and the task execution interval is short, the next task scheduling time may arrive before the previous task scheduling is complete. In this case, ROMA Connect suspends new scheduling and waits until the previous scheduling is complete.
1. On the **Create Task** page, configure schedule information. ROMA Connect provides two methods for configuring a schedule:
    - **GUI configuration**  
Currently, only the simple periodic configuration is supported, for example, every few minutes, hours, or days.

**Figure 5-2** GUI configuration



**Table 5-3** Simple GUI configuration

| Parameter              | Description  |
|------------------------|--|
| Quartz Cron Expression | Select <b>Do not use</b> .   |
| Unit                   | Set the unit of the task execution period. The value can be <b>Minute</b> , <b>Hour</b> , <b>Day</b> , <b>Week</b> , or <b>Month</b> . |

| Parameter      | Description   |
|----------------|---|
| Period         | Set the task execution cycle. The value range varies depending on the value of <b>Unit</b> .<br>For example, if <b>Unit</b> is set to <b>Day</b> and <b>Period</b> is set to <b>1</b> , the data integration task is executed once a day. |
| Effective Time | Start time of a task.   |
| Description    | Enter the description of the schedule.  |

– **Using Quartz Cron expression**

The Quartz Cron expression supports flexible schedules. For example, a task can be executed every 15 minutes from 01:00 a.m. to 04:00 a.m. every day based on the settings of the Quartz Cron expression. Such a schedule cannot be implemented through simple GUI configuration, but using the Quartz Cron expression.

0 0/15 1-4 \* \* ?

**Figure 5-3** Quartz Cron expression

Schedule

Quartz Cron Expression Do not use **Use**

\* Expression Schedule ?

| Second | Minute | Hour | Day | Month | Week |
|--------|--------|------|-----|-------|------|
| 0      | 0/15   | 1-4  | *   | *     | ?    |

\* Effective Time 2021/03/12 14:36:42 X 📅

Description 0/1,024

**Table 5-4** Using Quartz Cron expression

| Parameter              | Description         |
|------------------------|---------------------|
| Quartz Cron Expression | Select <b>Use</b> . |

| Parameter           | Description  |
|---------------------|--|
| Expression Schedule | Specify the Quartz Cron expression for the schedule. Currently, ROMA Connect supports only minute-level schedules. The second in the expression is fixed to <b>0</b> . For details, see <a href="#">Appendix: Quartz Cron Expression Configuration</a> .<br><br>For example, if a task needs to be executed every 15 minutes from 01:00 a.m. to 04:00 a.m. every day, the Quartz Cron expression corresponding to the schedule is as follows:<br><code>0 0/15 1-4 * * ?</code> |
| Effective Time      | Start time of a task.  |
| Description         | Enter the description of the schedule.   |

- After configuring the task schedule, proceed with [Configuring Source Information](#).

## 5.3.2 Configuring Source Information

### Overview

This topic describes how to configure source information for a data integration task. Based on the source information, ROMA Connect integrates data, including the data source type, data format, and data range. The source information configuration varies depending on data source types.

| Data Source Types Supported by Scheduled Integration Tasks  |   | Data Source Types Supported by Real-Time Integration Tasks  |
|---|---|---|
| <ul style="list-style-type: none"> <li>• <a href="#">APIs</a></li> <li>• <a href="#">DB2</a></li> <li>• <a href="#">DWS</a></li> <li>• <a href="#">DM</a></li> <li>• <a href="#">FTP</a></li> <li>• <a href="#">Gauss100</a></li> <li>• <a href="#">HANA</a></li> <li>• <a href="#">LDAP</a></li> <li>• <a href="#">MySQL</a></li> <li>• <a href="#">MongoDB</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">MRS Hive</a></li> <li>• <a href="#">MRS HDFS</a></li> <li>• <a href="#">MRS HBase</a></li> <li>• <a href="#">OBS</a></li> <li>• <a href="#">Oracle</a></li> <li>• <a href="#">PostgreSQL</a></li> <li>• <a href="#">SAP</a></li> <li>• <a href="#">SNMP</a></li> <li>• <a href="#">SQL Server</a></li> <li>• <a href="#">GaussDB(for MySQL)</a></li> <li>• <a href="#">Custom</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">ActiveMQ</a></li> <li>• <a href="#">ArtemisMQ</a></li> <li>• <a href="#">DIS</a></li> <li>• <a href="#">HL7</a></li> <li>• <a href="#">IBM MQ</a></li> <li>• <a href="#">Kafka</a></li> <li>• <a href="#">MQS</a></li> <li>• <a href="#">MRS Kafka</a></li> <li>• <a href="#">RabbitMQ</a></li> <li>• <a href="#">RocketMQ</a></li> <li>• <a href="#">WebSocket</a></li> </ul> |



## APIs

If **Integration Mode** is set to **Scheduled**, APIs can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-5** API source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the API data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>API</b> .   |
| Data Source Name        | Select the API data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Paging                  | <p>This parameter specifies whether data is returned on multiple pages when ROMA Connect sends a request to the API data source to obtain data. Multiple data records can be returned for one API request.</p> <ul style="list-style-type: none"><li>• If <b>Paging</b> is enabled, all data that meets the conditions is displayed on multiple pages based on a fixed number of records on each page. Each time an integration task is executed, ROMA Connect sends multiple API requests to obtain all data. That is, each API request is sent to obtain data on one page.</li><li>• If <b>Paging</b> is disabled, ROMA Connect obtains all data that meets the conditions through one API request.</li></ul> |
| Page Number Field       | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter a page number field defined in the API data source, for example, <b>pageNo</b>. This parameter is carried when ROMA Connect sends an API request to the source to specify the number of the page from which data is to be obtained. <b>Value</b> indicates whether the page number starts from 0 or 1. Set <b>Value</b> based on the original definition of the API.</p> <p>The page number field must be configured in <b>Params</b> or <b>Body of Request Parameters</b>.</p>   |
| Page Size Field         | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter a page size field defined in the API data source, for example, <b>pageSize</b>. This parameter is carried when ROMA Connect sends an API request to the source to specify the maximum number of records on each page. Set the number of records on each page based on the original definition of the API.</p>   |

| Parameter                 | Description  |
|---------------------------|--|
| Maximum Number of Pages   | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>This parameter specifies the maximum number of pages that can be queried in each scheduled task, for example, 10. If the number of pages exceeds the specified value, the task is stopped. The value <b>0</b> indicates that no restriction applies.</p>   |
| Pagination End            | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Select the method to stop obtaining source data in pagination mode.</p> <ul style="list-style-type: none"><li>• <b>Empty page list:</b> If no data record is returned, ROMA Connect stops obtaining source data.</li><li>• <b>Number of records:</b> ROMA Connect compares the calculation result based on the number of requested pages and the page size with the total number of records, to determine whether data stops to be obtained.</li></ul>   |
| Pagination End Field Path | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter the path of the field in an API response, which is used to determine the end of pagination. In the API response, elements in different layers are separated by periods (.). For example, if element c in the {"a":{"b":{"c":"xxx"}}} response is the pagination end field, the pagination end field path is set to <b>a.b.c</b>.</p> <ul style="list-style-type: none"><li>• If <b>Pagination End</b> is set to <b>Empty page list</b>, set this parameter to the root path of the list field.</li><li>• If <b>Pagination End</b> is set to <b>Number of records</b>, set this parameter to the path of the total number of records.</li></ul> |
| Incremental Migration     | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>   |
| Start Time Field          | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Enter the start time field originally defined in the API data source, for example, <b>startTime</b>. This parameter is carried when ROMA Connect sends an API request to the source, indicating that data following the specified field will be obtained.</p> <p>The start time field and end time field must be both entered in <b>Params</b> or <b>Body</b> of the <b>Request Parameters</b>.</p>  |

| Parameter                    | Description   |
|------------------------------|---|
| End Time Field               | Mandatory when <b>Incremental Migration</b> is enabled.<br>Enter the end time field originally defined in the API data source, for example, <b>endTime</b> . This parameter is carried when ROMA Connect sends an API request to the source, indicating that data before the specified value will be obtained.  |
| Time Zone                    | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the API data source so that ROMA Connect can identify the data timestamps.  |
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.<br>Assume that <b>Start Time Field</b> is <b>startTime</b> , <b>End Time Field</b> is <b>endTime</b> , <b>Timestamp Initial Value</b> is <b>2020-11-01 12:00:00</b> , <b>Compensation Period</b> is <b>0</b> , and <b>Period Settings</b> is <b>Default</b> for incremental collection. If the first scheduling time of the task is 2020-11-01 13:00:00, the data collected for the first time is that the value of <b>startTime</b> is greater than or equal to 2020-11-01 12:00:00 and the value of <b>endTime</b> is less than or equal to 2020-11-01 13:00:00. For subsequent collection, the data collected each time is that the value of <b>startTime</b> is greater than or equal to the time when the task is successfully executed last time and the value of <b>endTime</b> is less than or equal to the time. Execution time of the current task. |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.   |

| Parameter                | Description   |
|--------------------------|---|
| Compensation Period (ms) | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p>   |
| Time Format              | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select a timestamp format, for example, <i>yyyy-MM-dd</i>.</p>  |
| Period Settings          | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the mode used for setting the time range for subsequent data integration after an incremental migration task is executed for the first time.</p> <ul style="list-style-type: none"><li>• <b>Default:</b> Data generated between the previous scheduling and current scheduling is integrated. When ROMA Connect obtains data from the source, it uses the triggering time of the two tasks as the start time and end time, respectively.</li><li>• <b>Custom:</b> The start time and end time are determined based on the configured period rules. This mode applies to common periodic tasks, for example, tasks executed once a day, a week, or a month.</li></ul> |
| Start Time Offset (Days) | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Default</b>.</p> <p>Set the number of days before the start time of data collection.</p> <p>If data generated at the source changes in real time, such as alarm data, you can collect the data by setting this parameter.</p> <p>Start time of data collection = Data source system time - Start time offset</p>   |
| Time Interval            | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>Select the time granularity. The value must be the same as the unit configured in the <b>task schedule</b> so that the new data can be overwritten. For example, if <b>Unit</b> is set to <b>Day</b> in a task schedule, set this parameter to <b>Day</b>, indicating that data is obtained once a day.</p>   |

| Parameter               | Description   |
|-------------------------|---|
| Period                  | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>Select the time period for obtaining source data. For example, if the task is executed once a day, <b>Time Interval</b> is set to <b>Day</b>, and <b>Period</b> is set to <b>Previous period</b>, data of the previous day is incrementally integrated each time. If <b>Period</b> is set to <b>Current period</b>, data of the current day is incrementally integrated each time.</p>  |
| Right Periodic Boundary | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>This parameter specifies whether the end time is included in the time range for obtaining source data.</p> <ul style="list-style-type: none"> <li>• <b>Closed interval:</b> The end time is included.</li> <li>• <b>Open interval:</b> The end time is not included.</li> </ul>   |
| Request Parameters      | <p>Construct the parameter definition of the API request, for example, the page number and page size fields must be carried in <b>Params</b> or <b>Body</b>. Set this parameter based on the definition of the API data source.</p>   |
| Parse                   | <p>If <b>Paging</b> is enabled, <b>Parse</b> is set to <b>Yes</b> by default and cannot be changed.</p> <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"> <li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li> <li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li> </ul> |
| Response Type           | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>Select the format that will be used for the response of an API request. The value can be <b>JSON</b> or <b>XML</b>. Ensure that the format is the same as the actual response format of the API.</p>   |
| Data Root Field         | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON or XML format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</p>  |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each underlying key-value data element that is obtained from the source in JSON or XML format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the response.</li> <li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON or XML format does not contain arrays:

For example, in the following JSON data (similar to XML data), the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
  - **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
  - **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON or XML format contains arrays:
- For example, in the following JSON data (similar to XML data), the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**,

respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding data in JSON or XML format that contains arrays is used as an example. The following describes the configuration when the destination is API:

- In the example of pagination configuration. **pageNo** and **pageSize** are the pagination parameters of the API and need to be added to the **Request Parameters**.

**Figure 5-4** API pagination configuration example

\* Page Number Field ?  Value

\* Page Size Field ?  Records

Maximum Number of Pages  ?

Pagination End

Pagination End Field Path

\* Incremental Migration

Request Parameters

| Key                                   | Value                | Operation |
|---------------------------------------|----------------------|-----------|
| <input type="text" value="pageNo"/>   | <input type="text"/> | Delete    |
| <input type="text" value="pageSize"/> | <input type="text"/> | Delete    |

- In the example of incremental migration configuration, **startTime** and **endTime** are the time parameters of the API and need to be added to the **Request Parameters**.

**Figure 5-5** API incremental migration configuration example

\* Start Time Field ?

\* End Time Field ?

\* Time Zone

\* Timestamp Initial Value ?

\* Compensation Period (ms) ?

\* Time Format

\* Period Settings

Request Parameters

| Key                                    | Value                |
|--|----------------------|
| <input type="text" value="startTime"/> | <input type="text"/> |
| <input type="text" value="endTime"/>   | <input type="text"/> |

- In the example of metadata configuration, **Data Root Field** is set to **a**.



**Figure 5-6** API metadata configuration example

Parse: Yes

Response Type: JSON

Data Root Field: a

Metadata Table:

| Alias | Type    | Parsing Path | Operation     |
|-------|---------|--------------|---------------|
| c     | Integer | b[0].c       | Edit   Delete |
| d     | String  | b[0].d       | Edit   Delete |

Add

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## ActiveMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, ActiveMQ can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-6** ActiveMQ information at the source

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the ActiveMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>ActiveMQ</b> .  |
| Data Source Name        | Select the ActiveMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Destination Type        | Select the message transfer model of the ActiveMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name        | Enter the name of an existing topic or queue from which data is obtained.   |
| Parse                   | <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"> <li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li> <li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li> </ul> |

| Parameter       | Description   |
|-----------------|---|
| Data Root Field | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .   |
| Metadata        | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is ActiveMQ:

**Figure 5-7** ActiveMQ configuration example

The screenshot shows a configuration interface with the following fields and values:

- Destination Type:** Topic
- Destination Name:** doctest
- Parse:** Yes
- Data Root Field:** a
- Metadata Table:**

| Alias | Type    | Parsing Path |
|-------|---------|--------------|
| c     | Integer | b.c          |
| d     | String  | b.d          |

2. After configuring the source information, proceed with **Configuring Destination Information**.

## ArtemisMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, ArtemisMQ can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-7** ArtemisMQ information at the source

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the ArtemisMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>ArtemisMQ</b> .   |
| Data Source Name        | Select the ArtemisMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Destination Type        | Select the message transfer model of the ArtemisMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .   |
| Destination Name        | Enter the name of an existing topic or queue from which data is obtained.   |
| Parse                   | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| Data Root Field         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .                                   |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
  - **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
  - **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The configuration when the source is ArtemisMQ is similar to that when the source is ActiveMQ. For details, see [ActiveMQ configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## DB2

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, DB2 can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-8** DB2 source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the DB2 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>DB2</b> .  |

| Parameter             | Description   |
|-----------------------|---|
| Data Source Name      | Select the DB2 data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the DB2 data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize, such as the ID.</p>   |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the DB2 data source so that ROMA Connect can identify the data timestamps.</p>   |

| Parameter                    | Description  |
|------------------------------|--|
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type as the timestamp, which can be used to check whether a data row meets incremental integration conditions. If the entered values of <b>Timestamp</b> field and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.</p>   |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>   |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata            | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration



- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is DB2 is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## DWS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, DWS can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-9** DWS source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the DWS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>DWS</b> .   |
| Data Source Name        | Select the DWS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the DWS data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |

| Parameter                    | Description   |
|------------------------------|---|
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the DWS data source so that ROMA Connect can identify the data timestamps.</p>   |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>  |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is DWS is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## DIS

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, DIS can be selected as the source data source type.

1. On the **Create Task** page, configure source information.

**Table 5-10** DIS source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the DIS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>DIS</b> .   |
| Data Source Name        | Select the DIS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Parse                   | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| Data Root Field         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path in Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .  |

| Parameter | Description  |
|-----------|--|
| Data Type | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> .<br>Select the format of data obtained from the DIS data source. The data format must be the same as the actual data format stored in DIS.  |
| Metadata  | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> .<br>This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |
| Time Zone | Select the time zone used by the DIS data source so that ROMA Connect can identify the data timestamps.  |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The configuration when the source is DIS is similar to that when the source is ActiveMQ. For details, see [ActiveMQ configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## DM

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, DM can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-11** DM source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the DM data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>DM</b> .   |
| Data Source Name        | Select the DM data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>Example: <b>select string_col,number_col from zyw_test where time_col &gt;= <i>begin</i> and time_col &lt; <i>end</i></b></p> <ul style="list-style-type: none"><li>• <i>begin</i> indicates the time the task was last executed (do not change the format).</li><li>• <i>end</i> indicates the time when the current task starts execution (do not change the format).</li></ul> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table in the DM data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize, such as the ID.</p>   |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the DM data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type as the timestamp, which can be used to check whether a data row meets incremental integration conditions. If the entered values of <b>Timestamp</b> field and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.</p>                    |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |



| Parameter         | Description  |
|-------------------|--|
| Extended Metadata | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"> <li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li> <li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li> <li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is DM is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## FTP

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, FTP can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-12** FTP source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the FTP data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>FTP</b> .  |
| Data Source Name        | Select the FTP data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| File Path               | Enter the path of the folder to be accessed on the FTP server, for example, <b>/data/FDI</b> .   |
| File Name               | Enter the name of the data file from which data is to be obtained. If you do not specify this parameter, data of all files is obtained.  |
| File Name Encoding      | Select the encoding mode of the data file name.  |

| Parameter              | Description  |
|------------------------|--|
| Parse                  | <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| Maximum File Size      | <p>Enter the maximum size of the file to be obtained. If the size of a file exceeds the value you specify here, the file cannot be obtained.</p>   |
| File Content Encoding  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. Select the encoding format of the data file content.</p>   |
| File Separator         | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. Enter the field separator for the data file to distinguish different fields in each row of data.</p>   |
| Space Format Character | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. Enter the space characters defined in the data file so that ROMA Connect can identify the space characters in the file content. For example, if the space character is defined as a period (.), periods are considered as spaces during data reading.</p>  |
| Skip File Header       | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies whether to skip the file header in the data file. The file header is the first line or several lines at the beginning of a file, which helps identify and distinguish the file content.</p>   |
| Skipped Header Lines   | <p>This parameter is mandatory only if <b>Skip File Header</b> is set to <b>Yes</b>. Enter the number of lines in the file header in a data file so that ROMA Connect can skip the header lines and identify the start line of the data in the file.</p>   |
| Migrated Data Quantity | <p>Set the amount of data to be obtained.</p> <ul style="list-style-type: none"><li>• If <b>Parse</b> is set to <b>Yes</b>, this parameter indicates the number of data rows to be parsed.</li><li>• If <b>Parse</b> is set to <b>No</b>, this parameter indicates the number of files to be read.</li></ul>   |

| Parameter | Description  |
|-----------|--|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each data field in a data file obtained from the source to be integrated to the destination. Metadata must be filled according to the sequence of the fields in the file.</p> <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> </ul> |

The following describes the configuration when the source is FTP. The **id**, **name**, and **info** fields are obtained from the FTP data source and need to be integrated to the destination.

**Figure 5-8** FTP configuration example

The screenshot shows a configuration form for an FTP source. The parameters are as follows:

- \* File Path**: /data/test
- File Name**: sourcedata.csv
- \* File Name Encoding**: UTF-8
- \* Parse**: Yes
- \* Maximum File Size**: 0MB to 200MB (slider set to approximately 150MB)
- \* File Content Encoding**: UTF-8
- \* File Separator**: ,
- Space Format Character**: (empty)
- \* Skip File Header**: No
- \* 迁移数量**: 15,000
- \* Metadata**:
 

| Alias | Type    |
|-------|---------|
| id    | Integer |
| name  | String  |
| info  | String  |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## Gauss100

[Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, Gauss100 can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-13** Gauss100 source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the Gauss100 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>Gauss100</b> .   |
| Data Source Name        | Select the Gauss100 data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Schema                  | Mandatory when <b>Insert SQL</b> is disabled.<br>Select a schema.  |
| Table                   | Mandatory when <b>Insert SQL</b> is disabled.<br>Select the data table from which data is to be obtained in the Gauss100 data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.  |

| Parameter                    | Description   |
|------------------------------|---|
| Field Sorting                | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the Gauss100 data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>  |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is Gauss100 is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## HANA

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, HANA can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-14** HANA source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the HANA data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>HANA</b> .  |
| Data Source Name        | Select the HANA data source that you configured in <a href="#">Connecting to Data Sources</a> .   |

| Parameter             | Description  |
|-----------------------|--|
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the HANA data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>   |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the HANA data source so that ROMA Connect can identify the data timestamps.</p>   |



| Parameter                    | Description   |
|------------------------------|---|
| Timestamp Field              | This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.   |
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task. |
| Filter                       | Mandatory when <b>Insert SQL</b> is disabled.<br>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.<br>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.   |

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is HANA is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## HL7

[Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, HL7 can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-15** HL7 source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the HL7 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>HL7</b> .   |
| Data Source Name        | Select the HL7 data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Encoding Format         | Select the encoding mode of data files in the HL7 data source. The value can be <b>UTF-8</b> or <b>GBK</b> .  |
| Metadata                | This parameter specifies the data fields in HL7 messages obtained from the source to be integrated to the destination. <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata.</li> <li>• <b>Parsing Path</b>: location of metadata in HL7 messages. For details, see the metadata path configuration description in the subsequent section.</li> </ul> |

### Description on Metadata Path Configuration

```
MSH|^~\&|hl7Integration|hl7Integration|||ADT^A01|||2.3|
EVN|A01|20191212155644
PID||PATID1234^5^M11||FN^Patrick^^^MR||19700101|1||xx Street^^NY^^Ox4DP|||||
NK1|1|FN^John^^^MR|Father||999-9999
NK1|2|MN^Georgie^^^MSS|Mother||999-9999
```

The metadata parsing path of HL7 messages must be set based on the Terser syntax specifications. In the preceding example HL7 message, each line represents an information segment. Each information segment starts with three uppercase letters, which are paragraph symbols of the information segment and are used to indicate content of the information segment. Information segments are separated by separators.

- |: field separator, which is used to divide information segments into different fields. Each field in an information segment is numbered starting from 1 (excluding paragraph symbols). The rest may be deduced by analogy.
- ^: component separator, which divides the field content into different components. In the fields that are divided into components, the position of a component is identified by a number, starting from 1. The rest may be deduced by analogy.

- ~: subcomponent separator, which is used to divide a component into subcomponents.

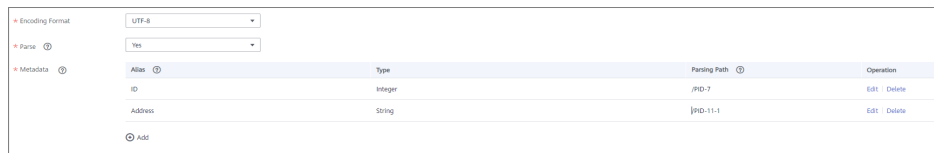
For example, in the PID information segment, if the field position of 19700101 is 7, the parsing path is **/PID-7**. If the field position of xx Street is 11 and the component position is 1, the parsing path is **/PID-11-1**.

For the information segments with the same paragraph symbol in the HL7 message, the repeated paragraph symbol is identified by adding a number enclosed by brackets after the paragraph symbol. In repeated paragraph symbols, the first is (0), the second is (1), and so on.

For example, in the NK1 information segment, "Father" is located in the first NK1 information segment, and a field location is 3, a parsing path of the NK1 information segment is **NK1(0)-3**. Similarly, the parsing path of Mother is **NK1(1)-3**.

Reading the 19700101 and xx Street fields in the preceding HL7 message is used as an example. The following describes the configuration when the source is HL7:

**Figure 5-9** HL7 configuration example



2. After configuring the source information, proceed with [Configuring Destination Information](#).

## IBM MQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, IBM MQ can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-16** IBM MQ information at the source

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the IBM MQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>IBM MQ</b> .  |
| Data Source Name        | Select the IBM MQ data source that you configured in <a href="#">Connecting to Data Sources</a> .   |

| Parameter        | Description   |
|------------------|---|
| Destination Type | Select the message transfer model of the IBM MQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name | Enter the name of an existing topic or queue from which data is obtained.   |
| Parse            | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>   |
| Data Root Field  | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .   |
| Metadata         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

```
}  
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
  - **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
  - **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{  
  "a": {  
    "b": [{  
      "c": "xx",  
      "d": "xx"  
    }],  
    {  
      "c": "yy",  
      "d": "yy"  
    }  
  }  
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The configuration when the source is IBM MQ is similar to that when the source is ActiveMQ. For details, see [ActiveMQ configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## Kafka

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, Kafka can be selected as the source data source. ROMA Connect MQS is a Kafka data source.

1. On the **Create Task** page, configure source information.

**Table 5-17** Kafka source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the Kafka data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>Kafka</b> .   |
| Data Source Name        | Select the Kafka data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Topic Name              | Select the name of the topic whose data is to be obtained.  |
| Parse                   | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| Data Root Field         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .                                   |
| Data Type               | Select the format of data obtained from the Kafka data source. The data format must be the same as the actual data format stored in Kafka. If <b>Parse</b> is set to <b>Yes</b> , you can select <b>JSON</b> or <b>XML</b> . If <b>Parse</b> is set to <b>No</b> , you can select <b>JSON</b> , <b>XML</b> , or <b>Binary file</b> .  |

| Parameter | Description  |
|-----------|--|
| Offset    | Select whether to integrate the earliest message data or the latest message data.  |
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |
| Time Zone | Select the time zone used by the Kafka data source so that ROMA Connect can identify the data timestamps.  |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

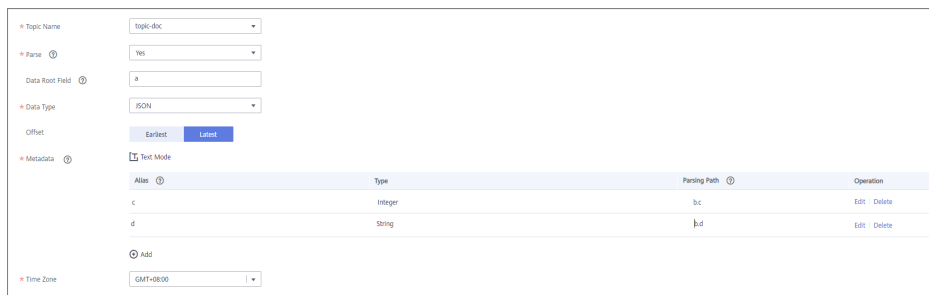
```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is Kafka:

**Figure 5-10** Kafka configuration example



2. After configuring the source information, proceed with **Configuring Destination Information**.

## LDAP

[Back to Overview](#)



If **Integration Mode** is set to **Scheduled**, LDAP can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-18** LDAP source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the LDAP data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>LDAP</b> .  |
| Data Source Name        | Select the LDAP data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| DN                      | This parameter specifies the distinguished name (DN) in the character string for connecting to the server, including Domain Component (DC), Common Name (CN), User ID (UID), and Organizational Unit (OU).<br>For example, if <b>DN</b> is set to <b>CN=test,OU=roma,DC=domainname</b> , it indicates the <b>test</b> object in the <b>roma</b> organization of the <b>domainname</b> domain.   |
| Filter Criteria         | Filter fields based on the LDAP syntax. For example, <code>( (uid=user.1*)&amp;(createTimestamp&gt;={{begin}})(createTimestamp&lt;={{end}}))</code> indicates that objects whose UID starts with user.1 within a specified period are queried. <ul style="list-style-type: none"><li>• <code>{{begin}}</code> indicates the time when the task was executed last time.</li><li>• <code>{{end}}</code> indicates the time when the task is executed this time.</li></ul> |
| Time Format             | Select a time format for the filter criterion.  |
| Timestamp Initial Value | This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.  |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter specifies each data field in a data file obtained from the source to be integrated to the destination. Metadata must be filled according to the sequence of the fields in the file.</p> <ul style="list-style-type: none"> <li>• <b>Alias:</b> user-defined metadata name.</li> <li>• <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> </ul> |

The following describes the configuration when the source is LDAP. The **id**, **name**, and **info** fields are obtained from the LDAP data source and need to be integrated to the destination.

**Figure 5-11** LDAP configuration example

The screenshot shows the LDAP configuration interface with the following fields and values:

- DN:** CN=test,OU=roma,DC=domainname
- Filter:** ((uid=user.1\*)&(createTimestamp>=\${b
- Filter Conditional Time Format:** Please select time format
- Timestamp Initial Value:** 2020/08/01 00:00:00
- Metadata Table:**

| Alias | Type    |
|-------|---------|
| id    | Integer |
| name  | String  |
| info  | String  |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MySQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MySQL can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-19** MySQL information at the source

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the MySQL data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>MySQL</b> .  |
| Data Source Name        | Select the MySQL data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the MySQL data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>  |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the MySQL data source so that ROMA Connect can identify the data timestamps.</p>   |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type as the timestamp, which can be used to check whether a data row meets data integration conditions. If the entered values of <b>Timestamp</b> field and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.</p>                           |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |

| Parameter         | Description  |
|-------------------|--|
| Extended Metadata | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

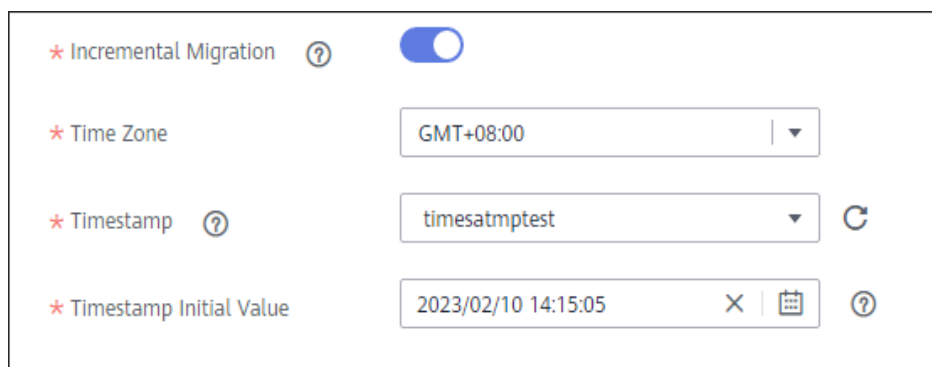
**NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The preceding JSON data that contains arrays is used as an example. The following describes the configuration when the source is MySQL:

- In the example of incremental migration configuration, the data table must contain a field of the DATE, TIME, or TIMESTAMP type as the timestamp field.

**Figure 5-12** MySQL incremental migration configuration example



- In the example of extended metadata configuration, the child elements **c** and **d** are obtained from the **desc** field in the data table.

**Figure 5-13** MySQL extended metadata configuration example

| Field Name | Type    | Parsing Path |
|------------|---------|--------------|
| desc       | integer | a.b.c        |
| desc       | string  | a.b.d        |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MongoDB

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MongoDB can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-20** MongoDB information at the source

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the MongoDB data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MongoDB</b> .   |
| Data Source Name        | Select the MongoDB data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Source Set              | Select the data set to be obtained from the MongoDB data source. (The data set is equivalent to a data table in a relational database.) Then, click <b>Select Fields in Set</b> and select only the column fields that you want to integrate and synchronize.   |
| Incremental Migration   | This parameter specifies whether only data generated in a specific period is integrated.<br><br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected. |
| Timestamp Field         | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.  |
| Time Zone               | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the MongoDB data source so that ROMA Connect can identify the data timestamps.  |
| Timestamp Initial Value | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |

| Parameter                    | Description   |
|------------------------------|---|
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>   |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms).</p> |

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is MongoDB is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MQS

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, MQS can be selected as the source data source. ROMA Connect MQS is an MQS data source.

1. On the **Create Task** page, configure source information.



**Table 5-21** MQS source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the MQS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MQS</b> .   |
| Data Source Name        | Select the MQS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Topic Name              | Select the name of the topic whose data is to be obtained.  |
| Parse                   | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| Data Root Field         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .                                   |
| Data Type               | Select the format of data obtained from the MQS data source. The data format must be the same as the actual data format stored in MQS. If <b>Parse</b> is set to <b>Yes</b> , select <b>JSON</b> or <b>XML</b> . If <b>Parse</b> is set to <b>No</b> , select <b>JSON</b> , <b>XML</b> , or <b>Binary file</b> .  |
| Offset                  | Select whether to integrate the earliest message data or the latest message data.   |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> <li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li> </ul> |
| Time Zone | Select the time zone used by the MQS data source so that ROMA Connect can identify the data timestamps.   |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c

and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is MQS:

**Figure 5-14** MQS configuration example

| Alias | Type    | Parsing Path | Operation     |
|-------|---------|--------------|---------------|
| c     | Integer | b.c          | Edit   Delete |
| d     | String  | b.d          | Edit   Delete |

2. After configuring the source information, proceed with **Configuring Destination Information**.

## MRS Hive

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MRS Hive can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-22** MRS Hive information at the source

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the MRS Hive data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MRS Hive</b> .   |
| Data Source Name        | Select the MRS Hive data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Database Name           | Select the database in the MRS Hive data source.<br><b>NOTE</b><br>You need to use a self-built database instead of the default database of MRS Hive.  |
| Table                   | Select the data table from which data is to be obtained in the MRS Hive data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.   |
| Incremental Migration   | This parameter specifies whether only data generated in a specific period is integrated.<br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected.<br>For subsequent scheduling, the data between the last successful collection time and the current time is collected. |
| Timestamp Field         | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.   |
| Time Zone               | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the MRS Hive data source so that ROMA Connect can identify the data timestamps.  |
| Timestamp Initial Value | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.  |

| Parameter                    | Description   |
|------------------------------|---|
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>   |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms).</p> |
| Filter                       | <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is MRS Hive is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MRS HDFS

[Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MRS HDFS can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-23** MRS HDFS source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the MRS HDFS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>MRS HDFS</b> .  |
| Data Source Name        | Select the MRS HDFS data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Separator               | Enter the field separator for the text data in the MRS HDFS data source. The separator is used to distinguish different fields in each row of data.   |
| Storage Subpath         | Enter the path of the data to be integrated in the <code>hdfs:///hacluster</code> directory of MRS HDFS.  |
| Storage Block Size (M)  | Select the size of data to be read each time ROMA Connect obtains data from the MRS HDFS data source.   |
| Storage Type            | Select the storage type of data in the MRS HDFS data source. The value must be the same as the actual data storage type of the MRS HDFS data source. The <b>TEXT</b> type is supported.   |
| Parse                   | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |

| Parameter | Description  |
|-----------|--|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each data field in text data obtained from the source to be integrated to the destination. Metadata must be filled according to the sequence of the fields in the file.</p> <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> </ul> |

The following describes the configuration when the source is MRS HDFS. The **id**, **name**, and **info** fields are obtained from the MRS HDFS data source and need to be integrated to the destination.

**Figure 5-15** MRS HDFS configuration example

The screenshot shows a configuration interface for MRS HDFS. It includes several fields with red asterisks indicating they are required:

- Separator**: A text input field containing a comma (,).
- Storage Subpath**: A text input field containing /data/test.
- Storage Block Size (M)**: A dropdown menu with 64 selected.
- Storage Type**: A dropdown menu with Text file selected.
- Parse**: A dropdown menu with Yes selected.
- Metadata**: A section with a question mark icon, containing a table of metadata fields:
 

| Alias | Type    |
|-------|---------|
| id    | Integer |
| name  | String  |
| info  | String  |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MRS HBase

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MRS HBase can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-24** MRS HBase information at the source

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the MRS HBase data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>MRS HBase</b> .  |
| Data Source Name        | Select the MRS HBase data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table                   | Select the data table in the MRS HBase data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.                    |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## MRS Kafka

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, MRS Kafka can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-25** MRS Kafka source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the MRS Kafka data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>MRS Kafka</b> .  |
| Data Source Name        | Select the MRS Kafka data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Topic Name              | Set this parameter to a topic that has been created in MRS Kafka.  |



| Parameter       | Description  |
|-----------------|--|
| Parse           | <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>   |
| Data Root Field | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path in Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Path Configuration</a>.</p>   |
| Data Type       | <p>Select the format of data obtained from the MRS Kafka data source, which must be consistent with that of the data stored in MRS Kafka.</p> <ul style="list-style-type: none"><li>• For <b>Parse</b> set to <b>Yes</b>: <b>JSON</b> or <b>XML</b></li><li>• For <b>Parse</b> set to <b>No</b>: <b>JSON</b>, <b>XML</b>, <b>Binary file</b></li></ul>   |
| Offset          | <p>Select whether to integrate the earliest message data or the latest message data.</p>   |
| Metadata        | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |
| Time Zone       | <p>Select the time zone used by the MRS Kafka data source so that ROMA Connect can identify the data timestamps.</p>   |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d

are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
  - **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
  - **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.

- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The configuration when the source is MRS Kafka is similar to that when the source is Kafka. For details, see [Kafka configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## OBS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, OBS can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-26** OBS source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the OBS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>OBS</b> .  |
| Data Source Name        | Select the OBS data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Path                    | Enter the path of the data files to be obtained in the OBS data source. The path cannot end with a slash (/) or be set to the root directory of an OBS bucket.                       |
| File Name Prefix        | Enter a file name prefix. This parameter is used together with <b>Time Format</b> to filter the data files to be integrated.   |
| Time Format             | Select the time format to be used in the file name. This parameter is used together with <b>File Name Prefix</b> to filter the data files to be integrated.                          |

| Parameter       | Description  |
|-----------------|--|
| Parse           | <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the files based on the default sequence of OBS. By default, the file with the latest time is parsed first and then integrated into the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul> |
| File Type       | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.<br/>Select the data file format of the OBS data source. The file format can be CSV, TXT, ZIP, XLS, or XLSX.</p>  |
| Field Separator | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.<br/>Enter the field separator for the data file to distinguish different fields in each row of data.</p>   |
| Encoding Format | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.<br/>Select the encoding mode of data files obtained from the OBS data source. The value can be <b>UTF-8</b> or <b>GBK</b>.</p>   |
| Skip Title      | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.<br/>This parameter specifies whether to skip the title lines in the data file. The title is the first line or several lines at the beginning of a file, which helps identify and distinguish the file content.</p>   |
| Title Lines     | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> and <b>Skip Title</b> is set to <b>Yes</b>.<br/>Enter the number of rows in the title information in the data file so that ROMA Connect can identify the start row of the data in the file.</p>   |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>. This parameter specifies each data field in a data file obtained from the source to be integrated to the destination. Metadata must be filled according to the sequence of the fields in the file.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data. The following types are supported:<ul style="list-style-type: none"><li>- <b>String</b></li><li>- <b>Double</b> (floating point number)</li><li>- <b>Date</b></li><li>- <b>Boolean</b></li><li>- <b>Long</b> (integer)</li></ul>Select <b>String</b> if you are not sure about the data type.</li></ul> |

The following describes the configuration when the source is OBS. The **id**, **name**, and **info** fields are obtained from the OBS data source and need to be integrated to the destination.

**Figure 5-16** OBS configuration example

| * Path            | <input type="text" value="data/test"/>   | ? |         |      |    |        |      |        |      |        |
|-------------------|--|---|---------|------|----|--------|------|--------|------|--------|
| File Name Prefix  | <input type="text" value="roma"/>  | ? |         |      |    |        |      |        |      |        |
| Time Format       | <input type="text" value="yyyy-MM-dd"/>  | ? |         |      |    |        |      |        |      |        |
| * Parse ?         | <input type="text" value="Yes"/>   |   |         |      |    |        |      |        |      |        |
| * File Type       | <input type="text" value="txt"/>   |   |         |      |    |        |      |        |      |        |
|                   | Parses txt files that comply with the csv format.  |   |         |      |    |        |      |        |      |        |
| * Field Separator | <input type="text" value=","/>   |   |         |      |    |        |      |        |      |        |
| * Encoding Format | <input type="text" value="UTF-8"/>   |   |         |      |    |        |      |        |      |        |
| * Skip Title      | <input type="text" value="No"/>  |   |         |      |    |        |      |        |      |        |
| * Metadata        | <input checked="" type="checkbox"/> Text Mode <table border="1"> <thead> <tr> <th>Alias ?</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>id</td> <td>Double</td> </tr> <tr> <td>name</td> <td>String</td> </tr> <tr> <td>info</td> <td>String</td> </tr> </tbody> </table> |   | Alias ? | Type | id | Double | name | String | info | String |
| Alias ?           | Type   |   |         |      |    |        |      |        |      |        |
| id                | Double   |   |         |      |    |        |      |        |      |        |
| name              | String   |   |         |      |    |        |      |        |      |        |
| info              | String   |   |         |      |    |        |      |        |      |        |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## Oracle

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, Oracle can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-27** Oracle source information

| Parameter | Description  |
|-----------|--|
| Instance  | Select the ROMA Connect instance that is being used. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the Oracle data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>Oracle</b> .  |
| Data Source Name        | Select the Oracle data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the Oracle data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>  |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration   | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |

| Parameter                    | Description   |
|------------------------------|---|
| Time Zone                    | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the Oracle data source so that ROMA Connect can identify the data timestamps.   |
| Timestamp Field              | This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.   |
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.   |
| Filter                       | Mandatory when <b>Insert SQL</b> is disabled.<br>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.<br>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.   |
| Extended Metadata            | Mandatory when <b>Insert SQL</b> is disabled.<br>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected. <ul style="list-style-type: none"> <li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li> <li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li> <li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li> </ul> |



### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is Oracle is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## PostgreSQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, PostgreSQL/openGauss can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-28** PostgreSQL/openGauss source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the PostgreSQL/openGauss data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>PostgreSQL/openGauss</b> .   |
| Data Source Name        | Select the PostgreSQL/openGauss data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the PostgreSQL/openGauss data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the PostgreSQL/openGauss data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>  |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |

| Parameter         | Description  |
|-------------------|--|
| Extended Metadata | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"> <li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li> <li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li> <li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  }
}
```

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is PostgreSQL/openGauss is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## RabbitMQ

[Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, RabbitMQ can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-29** RabbitMQ source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the RabbitMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>RabbitMQ</b> .   |
| Data Source Name        | Select the RabbitMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| New Queue Creation      | This parameter specifies whether to create a queue in the RabbitMQ data source. <ul style="list-style-type: none"><li>• If this parameter is set to <b>Yes</b>, a new queue is created and data is obtained from the new queue.</li><li>• If this parameter is set to <b>No</b>, data is obtained from the existing queue.</li></ul> |

| Parameter          | Description  |
|--------------------|--|
| Exchange Mode      | <p>This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b>.</p> <p>Select a routing mode for the exchange in RabbitMQ to forward messages to the new queue.</p> <ul style="list-style-type: none"><li>• <b>Direct</b>: If the routing key in a message fully matches the queue, the message will be forwarded to the queue.</li><li>• <b>Topic</b>: If the routing key in a message approximately matches the queue, the message will be forwarded to the queue.</li><li>• <b>Fanout</b>: All messages will be forwarded to the queue.</li><li>• <b>Headers</b>: If the Headers attribute of a message fully matches the queue, the message will be forwarded to the queue.</li></ul> |
| Exchange Name      | <p>This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b>.</p> <p>Enter the exchange name of the new queue in RabbitMQ.</p>  |
| Routing Key        | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Direct</b> or <b>Topic</b>.</p> <p>Enter the routing key of the new queue. RabbitMQ uses the routing key as a condition and forwards messages that meet the condition to the new queue.</p>   |
| Message Parameters | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Headers</b>.</p> <p>Enter the Headers key-value pair of the new queue. RabbitMQ uses Headers as a condition and forwards messages that meet the condition to the new queue.</p>   |
| Queue Name         | <p>Enter the name of the message queue whose data is to be obtained.</p> <ul style="list-style-type: none"><li>• If <b>New Queue Creation</b> is set to <b>Yes</b>, enter a new queue name.</li><li>• If <b>New Queue Creation</b> is set to <b>No</b>, enter the name of an existing queue in the RabbitMQ data source.</li></ul>   |
| Automatic Deletion | <p>This parameter specifies whether a queue will be automatically deleted if no client is connected.</p>   |
| Persistence        | <p>This parameter specifies whether messages in a queue are stored permanently.</p>  |

| Parameter | Description   |
|-----------|---|
| Parse     | <p>This parameter specifies whether ROMA Connect parses the obtained source data.</p> <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>  |
| Metadata  | <p>This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b>.</p> <p>This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
```

```

    "d": "xx"
  },
  {
    "c": "yy",
    "d": "yy"
  }
]
}

```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is RabbitMQ:

**Figure 5-17** RabbitMQ configuration example

The screenshot shows a configuration form for RabbitMQ. It includes several sections:

- \* New Queue Creation:** Radio buttons for 'Yes' and 'No'.
- \* Queue Name:** Text input field containing 'doctest'.
- \* Automatic Deletion:** Radio buttons for 'Yes' and 'No'.
- \* Persistency:** Radio buttons for 'Yes' and 'No'.
- \* Parse:** Dropdown menu set to 'Yes'.
- \* Metadata:** A table with columns 'Alias', 'Type', and 'Parsing Path'.
 

| Alias | Type    | Parsing Path |
|-------|---------|--------------|
| c     | Integer | a.b.c        |
| d     | String  | a.b.d        |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## RocketMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, RocketMQ can be selected as the source data source.

#### NOTE

The RocketMQ consumer group is named in the format of *task ID+\_consumer\_group*.

1. On the **Create Task** page, configure source information.

**Table 5-30** RocketMQ source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the RocketMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>RocketMQ</b> .  |



| Parameter        | Description  |
|------------------|--|
| Data Source Name | Select the RocketMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Topic Name       | Enter the topic character string or queue name.  |
| Parse            | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>  |
| Data Root Field  | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .  |
| Metadata         | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li><b>Alias</b>: user-defined metadata name.</li><li><b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li><b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

```
}
}
```

- Data in JSON format contains arrays:

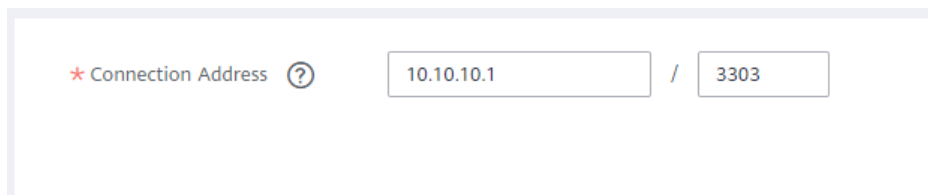
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

The following figure describes the configuration when the source is RocketMQ.

**Figure 5-18** RocketMQ configuration example



2. After configuring the source information, proceed with **Configuring Destination Information**.

## SAP

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, SAP can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-31** SAP source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the SAP data source belongs. Ensure that the integration application has been configured in <b>Connecting to Data Sources</b> . |

| Parameter          | Description   |
|--------------------|---|
| Data Source Type   | Select <b>SAP</b> .   |
| Data Source Name   | Select the SAP data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Connection Method  | Select the connection mode of the SAP data source. <ul style="list-style-type: none"> <li>• <b>Available Function</b>: specifies the functions that are opened by the SAP data source for remote calling.</li> <li>• <b>Table Connector</b>: connects to SAP data tables through a connector.</li> </ul>  |
| Available Function | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Available Function</b> .<br>Click <b>Query</b> . In the <b>Available Functions</b> dialog box, enter the function name and group name to query and obtain the list of available functions of the SAP data source. In the <b>Function List</b> area, select the function you want to use and click <b>OK</b> .<br><b>NOTE</b> <ul style="list-style-type: none"> <li>• The available functions can be queried on ROMA Connect only if the function that allows remote access has been enabled in the SAP data source.</li> <li>• The function name and group name support fuzzy search by prefix.</li> </ul> |
| Query Function     | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b> .<br>The default function is used generally. You can also select another query function as required, for example, <b>/SAPDS/RFC_READ_TABLE2</b> .<br><b>NOTICE</b><br>SAP does not support cross-table query because the <b>/SAPDS/RFC_READ_TABLE2</b> function can only be called to query one table.  |
| Table              | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b> .<br>Select a source data table. After selecting a data table, click <b>Select Table Field</b> to select specific table fields.   |
| Field Separator    | This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b> .<br>Enter a field separator to separate fields.  |

| Parameter             | Description   |
|-----------------------|---|
| Page Size             | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b>.</p> <p>Specify the maximum number of data records that can be read from the SAP server at a time, in 10,000.</p> <p><b>NOTICE</b><br/>A larger value indicates a higher read speed but may cause memory overflow, in which case the entire instance will fail to provide services. Retain the default value, or configure the parameter based on the actual service volume and instance specifications. In addition, to reduce the risk of memory overflow, set the number of concurrent tasks to be less than or equal to 5.</p> |
| Incremental Migration | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b>.</p> <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Partition Options     | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b>.</p> <p>Configure data read by partition to increase read speed. Data can be read in partitions by year, month, or day, or read without partitions.</p> <p>For example, if you want to read data from the first day to the 30th day of a month, set this parameter to <b>Day</b> and data is grouped into partitions by these 30 days monthly.</p> <p><b>NOTE</b><br/>Both full and incremental migration support partitioning.</p>  |
| Time Zone             | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled, or if <b>Partition Options</b> is set to <b>Day, Month, or Year</b>.</p> <p>Select the time zone used by the SAP data source so that ROMA Connect can identify the data timestamps.</p>   |

| Parameter               | Description   |
|-------------------------|---|
| Timestamp Field         | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled, or if <b>Partition Options</b> is set to <b>Day, Month, or Year</b>.</p> <p>Select a field of the DATE type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions. If the values of <b>Timestamp</b> and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.</p> <p><b>NOTICE</b><br/>SAP does not support incremental integration without the timestamp fields. That is, the table fields to be extracted must contain the timestamp fields.</p>  |
| Timestamp Initial Value | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled, or if <b>Partition Options</b> is set to <b>Day, Month, or Year</b>.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| End Timestamp           | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the end time of the data to be integrated. That is, only the data generated before this time point will be integrated.</p>   |
| Time Format             | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled, or if <b>Partition Options</b> is set to <b>Day, Month, or Year</b>.</p> <p>Select the time format to be used in the file name. This parameter is used together with <b>File Name Prefix</b> to filter the data files to be integrated. If the time format is selected and the file name prefix is not specified, files will not be filtered. After you select a time format, the task execution time will be converted into the selected time format.</p> <p>For example, if <b>File Name Prefix</b> is <b>test</b>, <b>Time Format</b> is <b>yyyyMMdd</b>, and the task execution time is 2021-01-14 10:00:00, data will be collected only from the files whose name prefix is <b>test20210114</b>.</p> |

| Parameter                | Description   |
|--------------------------|---|
| Compensation Period (ms) | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p> |
| Filter                   | <p>This parameter is mandatory only if <b>Connection Mode</b> is set to <b>Table Connector</b>.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |

### Description on Configuration of Table-type Request Parameters:

The following request parameter structure is used as an example. [] indicates the table-type data, and {} indicates a row of data in the table. There are three rows of data in a parameter table, and each row of data contains two parameters.

The key of parameter b is **a[i].b**, and the key of parameter c is **a[i].c**. i indicates the row number of the parameter in the parameter table. The value 0 indicates the first row, the value 1 indicates the second row, and so on.

```
a: [  
  {b: xx, c: xx},  
  {b: yy, c: yy},  
  {b: zz, c: zz},  
]
```

After you select an available function, only the parameters in the first row of a data table are displayed, that is, **a[0].b** and **a[0].c**. If you need to add the parameters of a row in the data table, add the parameters row by row in sequence. Ensure that the parameters of the previous row have been added. You are not allowed to add the parameters of only row 1 and row 3 without adding the parameters of row 2.

For example, in the preceding example, if you want to add the parameter in the second row, add **a[1].b** or **a[1].c** after **a[0].c**. To add the parameter in the third row, add at least one of parameters **a[1].b** and **a[1].c** in the second row after **a[0].c**, and then add parameter **a[2].b** or **a[2].c** in the third row.

**NOTICE**

The same parameter in the same row of data cannot be added repeatedly. Otherwise, the data integration task fails to be executed.

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## SNMP

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, SNMP can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-32** SNMP source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the SNMP data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>SNMP</b> .  |
| Data Source Name        | Select the SNMP data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Mode               | Select a data mode from the SNMP data source. <ul style="list-style-type: none"><li>• If you select <b>Row</b> for <b>Data Mode</b>, SNMP data is parsed by row.</li><li>• If you select <b>Column</b> for <b>Data Mode</b>, SNMP data is parsed by column.</li></ul> |
| Object Identifier       | This parameter is mandatory only if <b>Data Mode</b> is set to <b>Column</b> .<br>Enter the OID of the object to be obtained from the SNMP message. Separate multiple OIDs with commas (,).   |

| Parameter | Description   |
|-----------|---|
| Metadata  | <p>This parameter specifies the object (data field) to be integrated to the destination from the SNMP message data obtained from the source.</p> <ul style="list-style-type: none"> <li>• <b>Alias:</b> user-defined metadata name.</li> <li>• <b>Type:</b> data type of metadata.</li> <li>• <b>Parsing Path:</b> OID of metadata in the SNMP message. This parameter is mandatory only if <b>Data Mode</b> is set to <b>Row</b>.</li> </ul> |

The following describes the configuration when the source is SNMP:

- When **Data Model** is set to **Row**, **user** indicates obtaining system contact information and **location** indicates obtaining server location information.

**Figure 5-19** Configuration example when the data model is Row

|              |          |        |                    |
|--------------|----------|--------|--------------------|
| * Data Mode  | Row      |        |                    |
| * Metadata ? | Alias ?  | Type   | Parsing Path ?     |
|              | user     | String | .1.3.6.1.2.1.1.4.0 |
|              | location | String | .1.3.6.1.2.1.1.6.0 |

- When **Data Model** is set to **Column**, the **info** column contains the system contact information and server location information.

**Figure 5-20** Configuration example when the data model is Column

|                     |                                       |        |
|---------------------|---------------------------------------|--------|
| * Data Mode         | Column                                |        |
| * Object Identifier | .1.3.6.1.2.1.1.4.0,.1.3.6.1.2.1.1.6.0 |        |
| * Metadata          | Alias ?                               | Type   |
|                     | info                                  | String |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## SQL Server

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, SQL Server can be selected as the source data source.

1. On the **Create Task** page, configure source information.



**Table 5-33** SQL Server source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the SQL Server data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>SQL Server</b> .   |
| Data Source Name        | Select the SQL Server data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table in the SQL Server data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Incremental Migration        | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>  |
| Time Zone                    | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the SQL Server data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field              | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>  |
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p> |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>  |

| Parameter         | Description  |
|-------------------|--|
| Extended Metadata | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

 NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is SQL Server is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## GaussDB(for MySQL)

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, GaussDB(for MySQL) can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-34** GaussDB(for MySQL) source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the GaussDB(for MySQL) data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> . |
| Data Source Type        | Select <b>GaussDB(for MySQL)</b> .  |
| Data Source Name        | Select the GaussDB(for MySQL) data source that you configured in <a href="#">Connecting to Data Sources</a> .   |

| Parameter             | Description  |
|-----------------------|--|
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a <b>SELECT</b> statement containing <b>WHERE</b>. <b>INSERT</b>, <b>UPDATE</b>, <b>DELETE</b>, and <b>DROP</b> statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the GaussDB(for MySQL) data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>   |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For first-time scheduling, data between the initial timestamp and the specified timestamp is collected. The subsequent collection starts from the time when the last data record is imported in the previous collection to the specified timestamp.</p>   |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the GaussDB(for MySQL) data source so that ROMA Connect can identify the data timestamps.</p>   |
| Timestamp Field       | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a DATE field (down to the second) in the data table as the timestamp of the source data to determine whether the data meets the requirements of incremental integration.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.   |
| Filter                       | Mandatory when <b>Insert SQL</b> is disabled.<br>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.<br>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.   |
| Extended Metadata            | Mandatory when <b>Insert SQL</b> is disabled.<br>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected. <ul style="list-style-type: none"><li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
```

```
"d": "xx"
}
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The preceding JSON data that contains arrays is used as an example. The following describes the configuration for the GaussDB(for MySQL) destination.

- In the example of incremental migration configuration, the data table must contain a field of the DATE type as the timestamp field.

**Figure 5-21** Incremental migration configuration example of GaussDB(for MySQL)

The screenshot shows a configuration panel with the following settings:

- Incremental Migration:** Enabled (toggle switch).
- Time Zone:** GMT+08:00 (dropdown menu).
- Timestamp:** timesatmptest (dropdown menu).
- Timestamp Initial Value:** 2020/08/01 00:00:00 (text input with a calendar icon).
- Compensation Period (ms):** 100 (text input).

- In the example of extended metadata configuration, the child elements **c** and **d** are obtained from the **desc** field in the data table.

**Figure 5-22** GaussDB(for MySQL) extended metadata configuration example

|            |                   |             |                     |
|------------|-------------------|-------------|---------------------|
| Filter     | ⊕ Add Condition   |             |                     |
| metaData ? | <b>Field Name</b> | <b>Type</b> | <b>Parsing Path</b> |
|            | desc              | integer     | a.b.c               |
|            | desc              | string      | a.b.d               |
|            | ⊕ Add             |             |                     |

2. After configuring the source information, proceed with [Configuring Destination Information](#).

## WebSocket

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, WebSocket can be selected as the source data source.

1. On the **Create Task** page, configure source information.

**Table 5-35** WebSocket source information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Select the ROMA Connect instance that is being used.   |
| Integration Application | Select the integration application to which the WebSocket data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>WebSocket</b> .  |
| Data Source Name        | Select the WebSocket data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Root Field         | This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path in Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Path Configuration</a> . |
| Heartbeat Mechanism     | This parameter specifies whether the heartbeat mechanism is enabled between ROMA Connect and the WebSocket data source to ensure connection validity.  |



| Parameter                  | Description  |
|----------------------------|--|
| Heartbeat Message          | This parameter is mandatory only if <b>Heartbeat Mechanism</b> is set to <b>Yes</b> .<br>This parameter specifies the message content carried in heartbeat packets sent by ROMA Connect to the WebSocket data source.  |
| Heartbeat Sending Interval | This parameter is mandatory only if <b>Heartbeat Mechanism</b> is set to <b>Yes</b> .<br>This parameter specifies the interval for the ROMA Connect to send heartbeat packets to the WebSocket data source.  |
| Parse                      | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>  |
| Metadata                   | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> .<br>This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
  - **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
  - **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:
- For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is WebSocket:

**Figure 5-23** WebSocket configuration example

2. After configuring the source information, proceed with **Configuring Destination Information**.

## Custom

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, custom data sources can be selected at the source. ROMA Connect communicates with custom data sources through standard RESTful APIs.

1. On the **Create Task** page, configure source information.

**Table 5-36** Custom data source information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Select the ROMA Connect instance that is being used.  |
| Integration Application | Select the integration application to which the custom data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select a custom data source type.   |
| Data Source Name        | Select the custom data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Paging                  | <p>Specifies whether data is returned on multiple pages when ROMA Connect sends a request to the custom data source to obtain data. Multiple data records can be returned for one API request.</p> <ul style="list-style-type: none"> <li>• If <b>Paging</b> is enabled, all data that meets the conditions is displayed on multiple pages based on a fixed number of records on each page. ROMA Connect obtains data on one page each time and obtains all data by sending multiple requests.</li> <li>• If <b>Paging</b> is disabled, ROMA Connect obtains all data that meets the conditions through one API request.</li> </ul> |

| Parameter             | Description   |
|-----------------------|---|
| Start Page            | This parameter is mandatory only if <b>Paging</b> is enabled. Enter the start page number for paging. The value can be 0 or 1. Set this parameter based on the original definition of the API.  |
| Page Size             | This parameter is mandatory only if <b>Paging</b> is enabled. Enter the maximum number of data records on each page. Set this parameter based on the original definition of the API.  |
| Incremental Migration | This parameter specifies whether only data generated in a specific period is integrated.<br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.   |
| Time Zone             | Select the time zone used by the custom data source so that ROMA Connect can identify the data timestamps.  |
| Start Time            | Mandatory when <b>Incremental Migration</b> is enabled. This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.  |
| Parse                 | This parameter specifies whether ROMA Connect parses the obtained source data. <ul style="list-style-type: none"><li>• If you select <b>Yes</b>, ROMA Connect parses the obtained source data based on the configured parsing rules and then integrates the data to the destination.</li><li>• If you select <b>No</b>, ROMA Connect transparently transmits the obtained source data and integrates the data to the destination.</li></ul>   |
| Metadata              | This parameter is mandatory only if <b>Parse</b> is set to <b>Yes</b> . This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the response.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

| Parameter | Description  |
|-----------|--|
|           | In addition to the preceding parameters, different custom data sources define different reader parameters. Set the parameters based on the original definition specifications of the connector. You can locate the connector used by the custom data source on the <b>Assets</b> page of the ROMA Connect console and view the reader parameter definition of the connector. |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

The preceding JSON data that does not contain arrays is used as an example. The following figure describes the configuration when the source is a custom data source. In the example, the value of **key** is the reader parameter defined in the connector.

**Figure 5-24** Custom data source configuration example

| * Paging                | <input checked="" type="checkbox"/>  |              |      |              |   |         |       |   |        |       |
|-------------------------|--|--------------|------|--------------|---|---------|-------|---|--------|-------|
| * Start Page            | <input type="text" value="1"/>   |              |      |              |   |         |       |   |        |       |
| * Page Size             | <input type="text" value="10"/>  |              |      |              |   |         |       |   |        |       |
| * Incremental Migration | <input checked="" type="checkbox"/>  |              |      |              |   |         |       |   |        |       |
| * Time Zone             | <input type="text" value="GMT+08:00"/>   |              |      |              |   |         |       |   |        |       |
| * Start Time            | <input type="text" value="2020/08/01 00:00:00"/>   |              |      |              |   |         |       |   |        |       |
| * Parse                 | <input type="text" value="Yes"/>   |              |      |              |   |         |       |   |        |       |
| * key                   | <input type="text" value="test"/>  |              |      |              |   |         |       |   |        |       |
| * Metadata              | <table border="1"> <thead> <tr> <th>Alias</th> <th>Type</th> <th>Parsing Path</th> </tr> </thead> <tbody> <tr> <td>c</td> <td>Integer</td> <td>a.b.c</td> </tr> <tr> <td>d</td> <td>String</td> <td>a.b.d</td> </tr> </tbody> </table> | Alias        | Type | Parsing Path | c | Integer | a.b.c | d | String | a.b.d |
| Alias                   | Type   | Parsing Path |      |              |   |         |       |   |        |       |
| c                       | Integer  | a.b.c        |      |              |   |         |       |   |        |       |
| d                       | String   | a.b.d        |      |              |   |         |       |   |        |       |

2. After configuring the source information, proceed with **Configuring Destination Information**.

### 5.3.3 Configuring Destination Information

#### Overview

This topic describes how to configure destination information for a data integration task. Based on the destination information (including the data source and data storage), ROMA Connect writes data to the destination. The destination information configuration varies depending on data source types.

**NOTE**

During data migration, if a primary key conflict occurs at the destination, data is automatically updated based on the primary key.

| Data Source Types Supported by Both Real-Time and Scheduled Integration Tasks  |   | Data Source Types Supported by Only Scheduled Tasks  |
|--|---|--|
| <ul style="list-style-type: none"> <li>• <a href="#">APIs</a></li> <li>• <a href="#">ActiveMQ</a></li> <li>• <a href="#">ArtemisMQ</a></li> <li>• <a href="#">DB2</a></li> <li>• <a href="#">DIS</a></li> <li>• <a href="#">DWS</a></li> <li>• <a href="#">DM</a></li> <li>• <a href="#">Gauss100</a></li> <li>• <a href="#">HL7</a></li> <li>• <a href="#">HANA</a></li> <li>• <a href="#">IBM MQ</a></li> <li>• <a href="#">Kafka</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">MySQL</a></li> <li>• <a href="#">MongoDB</a></li> <li>• <a href="#">MQS</a></li> <li>• <a href="#">MRS Hive</a></li> <li>• <a href="#">MRS HDFS</a></li> <li>• <a href="#">MRS HBase</a></li> <li>• <a href="#">MRS Kafka</a></li> <li>• <a href="#">Oracle</a></li> <li>• <a href="#">PostgreSQL</a></li> <li>• <a href="#">Redis</a></li> <li>• <a href="#">RocketMQ</a></li> <li>• <a href="#">RabbitMQ</a></li> <li>• <a href="#">SQL Server</a></li> <li>• <a href="#">GaussDB(for MySQL)</a></li> <li>• <a href="#">Custom Data Sources</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">FTP</a></li> <li>• <a href="#">OBS</a></li> </ul> |

## APIs

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select API as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-37** API destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the API data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .                  |
| Data Source Type        | Select <b>API</b> .   |
| Data Source Name        | Select the API data source that you configured in <a href="#">Connecting to Data Sources</a> .  |

| Parameter          | Description  |
|--------------------|--|
| Request Parameters | <p>Construct the parameter definition of an API request. For example, the data to be integrated to the destination must be defined in <b>Body</b>. Set this parameter based on the definition of the API data source.</p> <ul style="list-style-type: none"> <li>• <b>Params</b>: parameters defined after the question mark (?) in the request URL. Only fixed values can be transferred. The method for setting <b>Params</b> is similar to that for setting <b>Body</b> in <b>form-data</b> format.</li> <li>• <b>Headers</b>: message headers of RESTful requests. Only fixed values can be transferred. The method for setting <b>Headers</b> is similar to that for setting <b>Body</b> in <b>form-data</b> format.</li> <li>• <b>Body</b>: bottom-layer parameters in the body of RESTful requests. This parameter and <b>Data Root Field</b> constitute the complete body of a request sent to the destination API. Source data is transferred to the destination API through parameters defined in <b>Body</b>. <b>Body</b> supports two modes: <b>form-data</b> and <b>raw</b>. For details, see <a href="#">Description on Body Parameter Configuration</a>.</li> </ul> |
| Data Root Field    | <p>This parameter specifies the path of upper-layer common fields in all parameters in the body sent to the destination in JSON format. <b>Data Root Field</b> and <b>Body</b> in <b>Request Parameters</b> form the request body sent to the destination API.</p> <p>For example, if the body parameter is {"c":"xx","d":"xx"} and <b>Data Root Field</b> is set to <b>a.b</b>, the encapsulated request data is {"a":{"b":{"c":"xx","d":"xx"}}}.</p>   |

### Description on Body Parameter Configuration

- form-data mode:

Set **Key** to the parameter name defined by the API data source and leave **Value** empty. The key will be used as the destination field name in mapping information to map and transfer the value of the source field.

Figure 5-25 form-data mode

| Key | Value | Operation |
|-----|-------|-----------|
| c   |       | Delete    |
| d   |       | Delete    |



– Raw mode:

The raw mode supports the JSON, Array, and nested JSON formats. Enter an example body sent to the destination API in JSON format. ROMA Connect replaces the parameter values in the example based on the mapping configuration, and finally transfers the source data to the destination. The following is an example body in raw mode:

■ JSON format:

```
{
  "id": 1,
  "name": "name1"
}
```

Enter the body in JSON format, leave **Data Root Field** empty, and set the field names in **Mapping Information**.

The screenshot shows the configuration interface for a request. Under 'Request Parameters', the 'Body' tab is active, and the 'raw' radio button is selected. The body content is a JSON object: `{ "id": 1, "name": "name1" }`. Below this, the 'Data Root Field' is an empty text box. The 'Mapping Information' section is visible below, showing a table with two columns: 'Destination Field' and 'Source Field'. The table contains two rows: 'id' mapped to 'id' and 'name' mapped to 'name'.

| Destination Field | Source Field |
|-------------------|--------------|
| id                | id           |
| name              | name         |

■ Array format:

```
{
  "record": [
    {
      "id": 1,
      "name": ""
    }
  ]
}
```

Set **Data Root Field** to the JSONArray object name, for example, **record**. Enter field names in mapping information.

Request Parameters

Params | Headers | **Body**

form-data  raw

```

{
  "record": [
    {
      "id": 1,
      "name": ""
    }
  ]
}
    
```

Data Root Field ?

Mapping Information

\* Mapping Rule Text Mode

Ensure that the field type of the source matches that of the destination. Otherwise, an error may occur during type conversion.

| Destination Field | Source Field |
|-------------------|--------------|
| id                | id           |
| name              | name         |

■ Nested JSON format:

```

{
  "startDate": "",
  "record": [
    {
      "id": 1,
      "name": ""
    }
  ]
}
    
```

Leave **Data Root Field** blank. In mapping information, set the json fields to the field names and set the jsonArray fields to specific paths, for example, **record[0].id**.

Request Parameters

Params | Headers | **Body**

form-data  raw

```

{
  "startDate": "",
  "record": [
    {
      "id": 1,
      "name": ""
    }
  ]
}
    
```

Data Root Field ?

Mapping Information

\* Mapping Rule Text Mode

Ensure that the field type of the source matches that of the destination. Otherwise, an error may occur during type conversion.

| Destination Field | Source Field |
|-------------------|--------------|
| startDate         | date         |
| record[0].id      | id           |
| record[0].name    | name         |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## ActiveMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select ActiveMQ as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-38** ActiveMQ destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.  |
| Integration Application | Select the integration application to which the ActiveMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>ActiveMQ</b> .   |
| Data Source Name        | Select the ActiveMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Destination Type        | Select the message transfer model of the ActiveMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .   |
| Destination Name        | Enter the name of a topic or queue to which to send data. Ensure that the topic or queue already exists.   |
| Metadata                | Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the destination is ActiveMQ:

**Figure 5-26** ActiveMQ configuration example

The screenshot shows a configuration form for ActiveMQ. It includes a dropdown for 'Destination Type' set to 'Topic', a text field for 'Destination Name' set to 'doctest', and a metadata table with columns for Alias, Type, Parsing Path, and Operation.

| Alias | Type    | Parsing Path | Operation     |
|-------|---------|--------------|---------------|
| c     | Integer | a.b.c        | Edit   Delete |
| d     | String  | a.b.d        | Edit   Delete |

Below the table is an 'Add' button.

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## ArtemisMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select ArtemisMQ as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-39** ArtemisMQ destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.  |
| Integration Application | Select the integration application to which the ArtemisMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>ArtemisMQ</b> .  |
| Data Source Name        | Select the ArtemisMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Destination Type        | Select the message transfer model of the ArtemisMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name        | Enter the name of a topic or queue to which to send data. Ensure that the topic or queue already exists.   |
| Metadata                | Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  }
}
```

The configuration when the destination is ArtemisMQ is similar to that when the destination is ActiveMQ. For details, see [ActiveMQ configuration example](#).

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## DB2

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select DB2 as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-40** DB2 destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the DB2 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .                  |
| Data Source Type        | Select <b>DB2</b> .   |
| Data Source Name        | Select the DB2 data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Table                   | Select the data table to which data will be written. Then, click <b>Select Table Field</b> and select only the column fields that you want to write.  |

| Parameter           | Description   |
|---------------------|---|
| Batch Number Field  | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## DIS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select DIS as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-41** DIS destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the DIS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .                  |
| Data Source Type        | Select <b>DIS</b> .   |
| Data Source Name        | Select the DIS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## DWS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select DWS as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-42** DWS destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the DWS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>DWS</b> .   |
| Data Source Name        | Select the DWS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Table                   | Select the data table to which data will be written. Then, click <b>Select Table Field</b> and select only the table fields that you want to write.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Clear Table             | This parameter specifies whether to clear the destination table each time when a schedule starts.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## DM

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select DM as the data source type at the destination.



1. On the **Create Task** page, configure destination information.

**Table 5-43** DM destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the DM data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>DM</b> .  |
| Data Source Name        | Select the DM data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table                   | Select the data table to which data will be written. Then, click <b>Select Table Field</b> and select only the column fields that you want to write.  |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## FTP

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, you can select FTP as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-44** FTP destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.  |
| Integration Application | Select the integration application to which the FTP data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>FTP</b> .  |
| Data Source Name        | Select the FTP data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| File Path               | Enter the path of the folder to be accessed on the FTP server, for example, <b>/data/FDI</b> .   |
| File Name Prefix        | Enter the prefix of the file name. This parameter is used together with <b>File Name Extension</b> to define the name of the file to be written to the FTP data source.  |
| File Name Extension     | Select a timestamp format for the file name extension. This parameter is used together with <b>File Name Prefix</b> to define the data file to be written to the FTP data source.  |
| File Content Type       | Select the content type of the data file to be written. Currently, <b>Text file</b> and <b>Binary file</b> are supported.  |
| File Type               | Select the format of the data file to be written. If <b>File Content Type</b> is set to <b>Text file</b> , <b>CSV</b> and <b>TXT</b> are available. If <b>File Content Type</b> is set to <b>Binary file</b> , <b>XLS</b> and <b>XLSX</b> are available. |
| File Name Encoding      | Select the encoding mode of the data file name.  |
| File Content Encoding   | This parameter is mandatory only if <b>File Content Type</b> is set to <b>Text</b> .<br>Select the encoding format of the data file content.   |
| File Separator          | This parameter is mandatory only if <b>File Content Type</b> is set to <b>Text</b> .<br>Enter the field separator for the data file to distinguish different fields in each row of data. By default, the fields are separated with commas (,).           |
| Space Format Character  | Define the space characters in the data file for data writing.   |

| Parameter       | Description   |
|-----------------|---|
| Write Mode      | Select the mode in which integration data is written to a file. <ul style="list-style-type: none"> <li>● <b>Truncate:</b> Delete the file, re-create a file, and write data to the new file.</li> <li>● <b>Append:</b> Incrementally write data to an existing file.</li> </ul>   |
| Add File Header | Determine whether to add a file header.   |
| File Header     | This parameter is mandatory only if <b>Add File Header</b> is set to <b>Yes</b> .<br>Enter the file header information. Use commas (,) to separate multiple file headers.   |
| Metadata        | Define the data fields to be written to the destination file. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"> <li>● <b>Alias:</b> user-defined metadata name.</li> <li>● <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> </ul> |

The following figure shows a configuration example when the destination is FTP. **id**, **name**, and **info** are the data fields to be written to the FTP data source.

**Figure 5-27** FTP configuration example

The screenshot displays the configuration interface for an FTP destination. It includes several input fields and dropdown menus for parameters such as File Path, File Name Prefix, File Name Extension, File Content Type, File Type, File Name Encoding, File Content Encoding, File Separator, Space Format Character, Write Mode, and Add File Header. The Metadata section is a table with three columns: Alias, Type, and Operation. The table contains three rows of metadata records.

| * File Path             | <input type="text" value="/data/test"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
|-------------------------|---|---|------|-----------|----|---------|---|------|--------|---|------|--------|---|
| * File Name Prefix      | <input type="text" value="roma"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Name Extension   | <input type="text" value="yyyyMMdd"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Content Type     | <input type="text" value="Text"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Type             | <input type="text" value="CSV"/>  |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Name Encoding    | <input type="text" value="UTF-8"/>  |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Content Encoding | <input type="text" value="UTF-8"/>  |   |      |           |    |         |   |      |        |   |      |        |   |
| * File Separator        | <input type="text" value=","/>  |   |      |           |    |         |   |      |        |   |      |        |   |
| Space Format Character  | <input type="text"/>  |   |      |           |    |         |   |      |        |   |      |        |   |
| * Write Mode            | <input type="text" value="Append"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
| * Add File Header       | <input type="text" value="No"/>   |   |      |           |    |         |   |      |        |   |      |        |   |
| * Metadata              | <table border="1"> <thead> <tr> <th>Alias</th> <th>Type</th> <th>Operation</th> </tr> </thead> <tbody> <tr> <td>id</td> <td>Integer</td> <td><a href="#">Edit</a>   <a href="#">Delete</a></td> </tr> <tr> <td>name</td> <td>String</td> <td><a href="#">Edit</a>   <a href="#">Delete</a></td> </tr> <tr> <td>info</td> <td>String</td> <td><a href="#">Edit</a>   <a href="#">Delete</a></td> </tr> </tbody> </table> | Alias   | Type | Operation | id | Integer | <a href="#">Edit</a>   <a href="#">Delete</a> | name | String | <a href="#">Edit</a>   <a href="#">Delete</a> | info | String | <a href="#">Edit</a>   <a href="#">Delete</a> |
| Alias                   | Type  | Operation                                     |      |           |    |         |   |      |        |   |      |        |   |
| id                      | Integer   | <a href="#">Edit</a>   <a href="#">Delete</a> |      |           |    |         |   |      |        |   |      |        |   |
| name                    | String  | <a href="#">Edit</a>   <a href="#">Delete</a> |      |           |    |         |   |      |        |   |      |        |   |
| info                    | String  | <a href="#">Edit</a>   <a href="#">Delete</a> |      |           |    |         |   |      |        |   |      |        |   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## Gauss100

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select Gauss100 as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-45** Gauss100 destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the Gauss100 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>Gauss100</b> .  |
| Data Source Name        | Select the Gauss100 data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Schema                  | Select a data source schema.  |
| Table                   | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## HL7

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select HL7 as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-46** HL7 destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the HL7 data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>HL7</b> .   |
| Data Source Name        | Select the HL7 data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Encoding Format         | Select the encoding mode of data files in the HL7 data source. The value can be <b>UTF-8</b> or <b>GBK</b> .  |
| Message Encoding Type   | Select the message type of the data to be integrated. This parameter defines the purpose and usage of messages. Set this parameter based on the definition in the HL7 protocol.   |
| Trigger Event Type      | Select the event type corresponding to the message type. Set this parameter based on the definition in the HL7 protocol.  |
| Protocol Version        | Select the HL7 protocol version used by the HL7 data source.  |
| Metadata                | Define the data fields to be written to the destination HL7. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata.</li><li>• <b>Parsing Path</b>: location of metadata in HL7 messages. For details, see the metadata path configuration description in the subsequent section.</li></ul> |

### Description on Metadata Path Configuration

```
MSH|^~\&|hl7Integration|hl7Integration|||ADT^A01|||2.3|
EVN|A01|20191212155644
PID||PATID1234^5^M11||FN^Patrick^^MR||19700101|1||xx Street^^NY^^Ox4DP|||||
NK1|1|FN^John^^MR|Father||999-9999
NK1|2|MN^Georgie^^MSS|Mother||999-9999
```

The metadata parsing path of HL7 messages must be set based on the Terser syntax specifications. In the preceding example HL7 message, each line represents an information segment. Each information segment starts with three uppercase letters, which are paragraph symbols of the information segment and are used to indicate content of the information segment. Information segments are separated by separators.

- |: field separator, which is used to divide information segments into different fields. Each field in an information segment is numbered starting from 1 (excluding paragraph symbols). The rest may be deduced by analogy.
- ^: component separator, which divides the field content into different components. In the fields that are divided into components, the position of a component is identified by a number, starting from 1. The rest may be deduced by analogy.
- ~: subcomponent separator, which is used to divide a component into subcomponents.

For example, in the PID information segment, if the field position of 19700101 is 7, the parsing path is **/PID-7**. If the field position of xx Street is 11 and the component position is 1, the parsing path is **/PID-11-1**.

For the information segments with the same paragraph symbol in the HL7 message, the repeated paragraph symbol is identified by adding a number enclosed by brackets after the paragraph symbol. In repeated paragraph symbols, the first is (0), the second is (1), and so on.

For example, in the NK1 information segment, "Father" is located in the first NK1 information segment, and a field location is 3, a parsing path of the NK1 information segment is **NK1(0)-3**. Similarly, the parsing path of Mother is **NK1(1)-3**.

Writing the **19700101** and **xx Street** fields in the preceding HL7 message is used as an example. The following describes the configuration when the destination is HL7:

**Figure 5-28** HL7 configuration example

The screenshot shows a configuration interface for HL7. It includes several dropdown menus and a table for metadata configuration.

- Encoding Format:** UTF-8
- Message Encoding Type:** ADT
- Trigger Event Type:** A01
- Protocol Version:** 2.3
- Metadata Table:**

| Alias   | Type    | Parsing Path |
|---------|---------|--------------|
| ID      | Integer | /PID-7       |
| Address | String  | /PID-11-1    |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## HANA

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select HANA as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-47** HANA destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the HANA data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>HANA</b> .  |
| Data Source Name        | Select the HANA data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table                   | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## IBM MQ

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select IBM MQ as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-48** IBM MQ destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.  |
| Integration Application | Select the integration application to which the IBM MQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>IBM MQ</b> .   |
| Data Source Name        | Select the IBM MQ data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Destination Type        | Select the message transfer model of the IBM MQ data source. The value can be <b>Topic</b> or <b>Queue</b> .   |
| Destination Name        | Enter the name of a topic or queue to which to send data. Ensure that the topic or queue already exists.   |
| Metadata                | Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```



```
}  
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{  
  "a": {  
    "b": [{  
      "c": "xx",  
      "d": "xx"  
    }],  
    {  
      "c": "yy",  
      "d": "yy"  
    }  
  }  
}
```

The configuration when the destination is IBM MQ is similar to that when the destination is ActiveMQ. For details, see [ActiveMQ configuration example](#).

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## Kafka

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select Kafka as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-49** Kafka destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the Kafka data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .                |
| Data Source Type        | Select <b>Kafka</b> .   |
| Data Source Name        | Select the Kafka data source that you configured in <a href="#">Connecting to Data Sources</a> .  |

| Parameter  | Description   |
|------------|---|
| Topic Name | Select the name of the topic to which data is to be written.  |
| Key        | Enter the key value of a message so that the message will be stored in a specified partition. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner.   |
| Metadata   | <p>Define the data fields to be written to the destination Kafka. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"> <li>• <b>Alias:</b> user-defined metadata name.</li> <li>• <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> </ul> |

The following figure shows a configuration example when the destination is Kafka. **id**, **name**, and **info** are the data fields to be written to the Kafka data source.

**Figure 5-29** Kafka configuration example

The screenshot shows a configuration form for Kafka. It includes the following elements:

- Topic Name:** A text input field containing "topic-doc".
- Key:** A text input field with the placeholder "Enter a key".
- Metadata:** A section with a "Text Mode" icon and a table of metadata fields.

| Alias | Type    | Operation   |
|-------|---------|-------------|
| id    | Integer | Edit Delete |
| name  | String  | Edit Delete |
| info  | String  | Edit Delete |

The structure of the message written to Kafka is {"id": "xx", "name": "yy", "info": "zz"}, where xx, yy, and zz are the data values transferred from the source.

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MySQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MySQL as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-50** MySQL destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the MySQL data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MySQL</b> .   |
| Data Source Name        | Select the MySQL data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Table                   | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Clear Table             | When this function is enabled, the destination table is cleared before tasks are scheduled.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MongoDB

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MongoDB as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-51** MongoDB destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.                                  |
| Integration Application | Select the integration application to which the MongoDB data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>MongoDB</b> .  |
| Data Source Name        | Select the MongoDB data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Destination Set         | Select the data set to be written to the MongoDB data source. (The data set is equivalent to a data table in a relational database.) Then, click <b>Select Fields in Set</b> and select only the column fields that you want to write. |
| Upsert                  | This parameter indicates whether to update or insert data to the destination, that is, whether to directly updating existing data fields in the data set on the destination.   |
| Upsert key              | This parameter is mandatory only if <b>Upsert</b> is enabled. Select the data field to be upserted.  |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MQS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MQS as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-52** MQS destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the MQS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MQS</b> .   |
| Data Source Name        | Select the MQS data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Topic Name              | Select the name of the topic to which data is to be written.  |
| Key                     | Enter the key value of a message so that the message will be stored in a specified partition. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner. |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MRS Hive

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MRS Hive as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

#### NOTE

If the source data field contains special characters `\r`, `\n`, and `\01`, ROMA Connect deletes these special characters and then writes the data to MRS Hive.

**Table 5-53** MRS Hive destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the MRS Hive data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .             |
| Data Source Type        | Select <b>MRS Hive</b> .  |

| Parameter        | Description   |
|------------------|---|
| Data Source Name | Select the MRS Hive data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Database Name    | Select the database to which data will be written.<br><b>NOTE</b><br>You need to use a self-built database instead of the default database of MRS Hive.   |
| Table            | Select the data table to which data will be written.  |
| Separator        | Enter the field separator for the text data in the MRS Hive data source. The separator is used to distinguish different fields in each row of data.   |
| Write Mode       | Select the mode in which integration data is written to a data table. <ul style="list-style-type: none"><li>• <b>Truncate:</b> Delete all data from the destination data table and then write data to the table.</li><li>• <b>Append:</b> Incrementally write data to an existing data table.</li></ul> |
| Storage Type     | Select <b>RCFile</b> or <b>Text file</b> as the storage type for data to be written to the MRS Hive data source.  |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MRS HDFS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MRS HDFS as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

#### NOTE

If the source data field contains special characters `\r`, `\n`, and `\01`, ROMA Connect deletes these special characters and then writes the data to MRS HDFS.

**Table 5-54** MRS HDFS destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the MRS HDFS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>MRS HDFS</b> .  |
| Data Source Name        | Select the MRS HDFS data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Separator               | Enter the field separator for the text data in the MRS HDFS data source. The separator is used to distinguish different fields in each row of data.   |
| Storage Subpath         | Enter the path of the data to be integrated in the hdfs:///hacluster directory of MRS HDFS.   |
| Storage Block Size (M)  | Select the size of data to be written each time ROMA Connect writes data to the MRS HDFS data source.   |
| Storage Type            | Select <b>Text</b> as the data storage type of the MRS HDFS data source.  |
| Metadata                | Define the data fields to be written to the destination text data. Separate different data fields with delimiters. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> |

The following figure shows a configuration example when the destination is MRS HDFS. **id**, **name**, and **info** are the data fields to be written to the MRS HDFS data source.

**Figure 5-30** MRS HDFS configuration example

| * Separator <span>?</span>       | <input type="text" value="."/>   |                      |      |    |         |      |        |      |        |
|----------------------------------|--|----------------------|------|----|---------|------|--------|------|--------|
| * Storage Subpath <span>?</span> | <input type="text" value="/data/test"/>  |                      |      |    |         |      |        |      |        |
| * Storage Block Size (M)         | <input type="text" value="64"/>  |                      |      |    |         |      |        |      |        |
| * Storage Type                   | <input type="text" value="Text"/>  |                      |      |    |         |      |        |      |        |
| * Metadata <span>?</span>        | <table border="1"> <thead> <tr> <th>Alias <span>?</span></th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>id</td> <td>Integer</td> </tr> <tr> <td>name</td> <td>String</td> </tr> <tr> <td>info</td> <td>String</td> </tr> </tbody> </table> | Alias <span>?</span> | Type | id | Integer | name | String | info | String |
| Alias <span>?</span>             | Type   |                      |      |    |         |      |        |      |        |
| id                               | Integer  |                      |      |    |         |      |        |      |        |
| name                             | String   |                      |      |    |         |      |        |      |        |
| info                             | String   |                      |      |    |         |      |        |      |        |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MRS HBase

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MRS HBase as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

#### NOTE

If the source data field contains special characters \r, \n, and \01, ROMA Connect deletes these special characters and then writes the data to MRS HBase.

**Table 5-55** MRS HBase destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the MRS HBase data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .            |
| Data Source Type        | Select <b>MRS HBase</b> .   |
| Data Source Name        | Select the MRS HBase data source that you configured in <a href="#">Connecting to Data Sources</a> .  |



| Parameter     | Description   |
|---------------|---|
| Table         | Select the data table to which data will be written.  |
| Column Family | Define the data column fields to be written to the destination data table. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.<br><b>Field Name:</b> user-defined name of a field in a data column. |

The following figure shows a configuration example when the destination is MRS HBase. **id**, **name**, and **info** are the data fields to be written to the MRS HBase data source.

**Figure 5-31** MRS HBase configuration example

\* Table

\* Column Family

| Field Name                        |
|-----------------------------------|
| <input type="text" value="id"/>   |
| <input type="text" value="name"/> |
| <input type="text" value="info"/> |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## MRS Kafka

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select MRS Kafka as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-56** MRS Kafka destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the MRS Kafka data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .  |
| Data Source Type        | Select <b>MRS Kafka</b> .   |
| Data Source Name        | Select the MRS Kafka data source that you configured in <a href="#">Connecting to Data Sources</a> .  |
| Topic Name              | Set this parameter to a topic that has been created in MRS Kafka.   |
| Key                     | Enter the key value of a message so that the message will be stored in a specified partition. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner.   |
| Metadata                | Define the data fields to be written to the destination Kafka. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> |

The configuration when the destination is MRS Kafka is similar to that when the destination is Kafka. For details, see [Kafka configuration example](#).

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## OBS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, you can select OBS as the data source type at the source.

1. On the **Create Task** page, configure destination information.

**Table 5-57** OBS destination information

| Parameter               | Description  |
|-------------------------|--|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.                |
| Integration Application | Select the integration application to which the OBS data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .                                 |
| Data Source Type        | Select <b>OBS</b> .  |
| Data Source Name        | Select the OBS data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Object Type             | Select the type of the data file to be written to the OBS data source. Currently, <b>Text file</b> and <b>Binary file</b> are supported.   |
| Encoding Format         | This parameter is mandatory only if <b>Object Type</b> is set to <b>Text file</b> .<br>Select the encoding mode of the data file to be written to the OBS data source. The value can be <b>UTF-8</b> or <b>GBK</b> . |
| Path                    | Enter the path of the data files to be written to in the OBS data source. The path cannot end with a slash (/) or be set to the root directory of an OBS bucket.   |
| File Name Prefix        | Enter the prefix of the file name. This parameter is used together with <b>Time Format</b> to define the name of the file to be written to the OBS data source.  |
| Time Format             | Select the time format to be used in the file name. This parameter is used together with <b>File Name Prefix</b> to define the data file to be written to the OBS data source.                                       |
| File Type               | Select the format of the data file to be written to the OBS data source. A text file can be in TXT or CSV format, and a binary file can be in XLS or XLSX format.  |
| Advanced Attributes     | This parameter is mandatory only if <b>File Type</b> is set to <b>csv</b> .<br>Select whether to configure the advanced properties of the file.  |
| Newline                 | This parameter is mandatory only if <b>Advanced Attributes</b> is set to <b>Enable</b> .<br>Enter a newline character in the file content to distinguish different data lines in the file.                           |

| Parameter           | Description   |
|---------------------|---|
| Enclosure Character | <p>This parameter is mandatory only if <b>Advanced Attributes</b> is set to <b>Enable</b>.</p> <p>If you select <b>Use</b>, each data field in the data file is enclosed by double quotation marks (""). If a data field contains the same symbol as a separator or newline character, the field will not be split into two fields. For example, if source data contains a data field <b>aa bb</b> and the vertical bar ( ) is set as the separator when the source data is integrated to the destination data file, the field is <b>aa bb</b> in the destination data file and will not be split into <b>aa</b> and <b>bb</b>.</p>   |
| Field Separator     | <p>This parameter is mandatory only if <b>File Type</b> is set to <b>txt</b> or <b>Advanced Attributes</b> is set to <b>Enable</b>.</p> <p>Enter the field separator for the file contents to distinguish different fields in each row of data.</p>   |
| Add File Header     | <p>Determine whether to add a file header to the data file to be written. The file header is the first line or several lines at the beginning of a file, which helps identify and distinguish the file content.</p>   |
| File Header         | <p>This parameter is mandatory only if <b>Add File Header</b> is set to <b>Yes</b>.</p> <p>Enter the file header information. Use commas (,) to separate multiple file headers.</p>   |
| Metadata            | <p>Define the data fields to be written to the destination file. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.<br/>The following types are supported:<ul style="list-style-type: none"><li>- <b>String</b></li><li>- <b>Double</b> (floating point number)</li><li>- <b>Date</b></li><li>- <b>Boolean</b></li><li>- <b>Long</b> (integer)</li></ul></li></ul> <p>Select <b>String</b> if you are not sure about the data type.</p> |

The following figure shows a configuration example when the destination is OBS. **id**, **name**, and **info** are the data fields to be written to the OBS data source.

**Figure 5-32** OBS configuration example

| * Object Type        | Text file  |         |      |    |        |      |        |      |        |
|----------------------|--|---------|------|----|--------|------|--------|------|--------|
| * Encoding Format    | UTF-8  |         |      |    |        |      |        |      |        |
| * Path ?             | data/test  |         |      |    |        |      |        |      |        |
| * File Name Prefix ? | roma   |         |      |    |        |      |        |      |        |
| * Time Format ?      | yyyy-MM-dd   |         |      |    |        |      |        |      |        |
| * File Type          | txt  |         |      |    |        |      |        |      |        |
| * Field Separator    | ,  |         |      |    |        |      |        |      |        |
| * Add File Header    | No   |         |      |    |        |      |        |      |        |
| * Metadata ?         | <div style="border: 1px solid #ccc; padding: 5px;"> <div style="display: flex; align-items: center;"> <span style="font-size: 1.2em; margin-right: 5px;">T</span> Text Mode         </div> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr style="background-color: #f2f2f2;"> <th style="text-align: left; padding: 2px;">Alias ?</th> <th style="text-align: left; padding: 2px;">Type</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">id</td> <td style="padding: 2px;">Double</td> </tr> <tr> <td style="padding: 2px;">name</td> <td style="padding: 2px;">String</td> </tr> <tr> <td style="padding: 2px;">info</td> <td style="padding: 2px;">String</td> </tr> </tbody> </table> </div> | Alias ? | Type | id | Double | name | String | info | String |
| Alias ?              | Type   |         |      |    |        |      |        |      |        |
| id                   | Double   |         |      |    |        |      |        |      |        |
| name                 | String   |         |      |    |        |      |        |      |        |
| info                 | String   |         |      |    |        |      |        |      |        |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## Oracle

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select Oracle as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-58** Oracle destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
| Integration Application | Select the integration application to which the Oracle data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .               |

| Parameter           | Description   |
|---------------------|---|
| Data Source Type    | Select <b>Oracle</b> .  |
| Data Source Name    | Select the Oracle data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table               | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.<br><b>NOTE</b><br>If the source field corresponding to the primary key field is empty, the record is discarded by default and no scheduling log error code is generated.  |
| Batch Number Field  | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Clear Table         | When this function is enabled, the destination table is cleared before tasks are scheduled.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## PostgreSQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, PostgreSQL/openGauss can be selected as the data source at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-59** PostgreSQL/openGauss destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the PostgreSQL/openGauss data source belongs.   |
| Data Source Type        | Select <b>PostgreSQL/openGauss</b> .  |
| Data Source Name        | Select the PostgreSQL/openGauss data source that has been configured.   |
| Table                   | Select the data table to which data is to be written and click <b>Select Table Field</b> to select the data column fields to be integrated.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format     | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Clear Table             | When this function is enabled, the destination table is cleared before tasks are scheduled.   |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## Redis

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select Redis as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-60** Redis destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description  |
|-------------------------|--|
| Integration Application | Select the integration application to which the Redis data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>Redis</b> .  |
| Data Source Name        | Select the Redis data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Key Prefix              | Enter the key name prefix of the data to be integrated in the Redis data source. The key name in the Redis data source consists of the key prefix, separator, and key suffix field. Each row of data is stored in the Redis as the key value. For details about the key format, see <a href="#">Description on Key and Value Formats</a> .   |
| Key Suffix              | Select a field whose value is unique in the source data as the key suffix. The key name in the Redis data source consists of the key prefix, separator, and key suffix field. In this way, each row of data can be integrated into different keys of the Redis data source.<br><br>If <b>Data Type</b> is set to <b>List</b> , <b>Set</b> , or <b>ZSet</b> , <b>Key Suffix</b> can be left blank. That is, only one key is generated based on the key prefix. In this case, all data rows are integrated to the same key of the Redis data source as elements. |
| Separator               | This parameter is mandatory only if <b>Key Suffix</b> is not left blank.<br><br>Enter the separator between <b>Key Prefix</b> and <b>Key Suffix</b> . The key name in the Redis data source consists of <b>Key Prefix</b> , <b>Separator</b> , and <b>Key Suffix</b> .   |
| Data Type               | Select the data type of the key in the Redis data source. Options are as follows: <ul style="list-style-type: none"><li>• String</li><li>• List</li><li>• Map</li><li>• Set</li><li>• ZSet</li></ul>   |



| Parameter           | Description  |
|---------------------|--|
| List Appending Mode | <p>This parameter is mandatory only if <b>Data Type</b> is set to <b>List</b>.</p> <p>Select the data appending mode for the key of the list type.</p> <ul style="list-style-type: none"><li>• <b>lpush</b>: The current data is inserted to the header of the list.</li><li>• <b>rpush</b>: The current data is inserted to the end of the list.</li></ul>  |
| sortColumn          | <p>This parameter is mandatory only if <b>Data Type</b> is set to <b>ZSet</b>.</p> <p>Select the source data field for sorting data elements.</p>  |
| Validity Period (s) | <p>Period for which a key in the Redis data source remains valid. The value <b>0</b> indicates that the key never expires.</p>   |
| Write Format        | <p>This parameter is mandatory only if <b>Data Type</b> is set to <b>String</b>, <b>List</b>, <b>Set</b>, or <b>ZSet</b>. If <b>Data Type</b> is set to <b>Map</b>, the JSON format is used by default.</p> <p>Select <b>JSON</b> or <b>CUSTOMIZE</b> as the format of the data to be written to the Redis data source.</p>  |
| Metadata            | <p>Define the format of the value written to be written to the destination key. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> <p>If <b>Write Format</b> is set to <b>JSON</b>, the metadata is stored as the key value in the Redis data source in JSON format. If <b>Write Format</b> is set to <b>CUSTOMIZE</b>, customize the format of the destination value. All metadata is combined with the prefix and suffix, and then stored as the key value in the Redis data source. For details about the value format, see <a href="#">Description on Key and Value Formats</a>.</p> |

### Description on key and value formats

Assume that at the destination: **Key Prefix** is set to **roma**; **Key Suffix** is set to the unique key **aaa** of the source data, which ensures that the key name is unique; **Separator** is set to the vertical bar (|), which will be used as a separator between key prefixes and key suffixes.

```
+-----+-----+
| aaa | bbb |
+-----+-----+
| 1 | x |
```

```
| 2 | y |
| 3 | z |
+-----+
```

- If **Data Type** is set to **String** and **Write Format** is set to **JSON**, the keys and values written to the Redis data source are shown in [Figure 5-33](#).

```
key      value
-----
roma|1   "{\"bbb\":\"x\",\"aaa\":1}"
roma|2   "{\"bbb\":\"y\",\"aaa\":2}"
roma|3   "{\"bbb\":\"z\",\"aaa\":3}"
```

**Figure 5-33** Metadata configuration (JSON)

| * Metadata  | Field Name | Type   | Operation |
|---|------------|--------|-----------|
|   | bbb        | string | Delete    |
|   | aaa        | string | Delete    |
| <input type="button" value="Synchronize Fields from Data Source"/> <input type="button" value="⊕ Add"/> |            |        |           |

- If **Data Type** is set to **String** and **Write Format** is set to **CUSTOMIZE**, the keys and values written to the Redis data source are shown in [Figure 5-34](#).

```
key      value
-----
roma|1   "bbb_x&aaa_1"
roma|2   "bbb_y&aaa_2"
roma|3   "bbb_z&aaa_3"
```

**Figure 5-34** Metadata configuration (CUSTOMIZE)

| * Metadata  | Field Name | Type   | Operation |
|---|------------|--------|-----------|
|   | bbb        | string | Delete    |
|   | aaa        | string | Delete    |
| <input type="button" value="Synchronize Fields from Data Source"/> <input type="button" value="⊕ Add"/> |            |        |           |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## RocketMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select RocketMQ as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-61** RocketMQ destination information

| Parameter | Description   |
|-----------|---|
| Instance  | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the RocketMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>RocketMQ</b> .  |
| Data Source Name        | Select the RocketMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Topic                   | Topic character string or queue name.   |
| Metadata                | <p>Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"> <li>• <b>Alias</b>: user-defined metadata name.</li> <li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> <li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",

```

```

    "d": "xx"
  },
  {
    "c": "yy",
    "d": "yy"
  }
]
}

```

The following figure describes the configuration when the destination is RocketMQ.

**Figure 5-35** RocketMQ configuration example

| Alias | Type              | Partition Path | Operation   |
|-------|-------------------|----------------|-------------|
| c     | Integer           | a.b.d          | Edit Delete |
| d     | String yyyy-MM-dd | b.b.d          | Edit Delete |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## RabbitMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select RabbitMQ as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-62** RabbitMQ destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the RabbitMQ data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>RabbitMQ</b> .  |
| Data Source Name        | Select the RabbitMQ data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| New Queue Creation      | Determine whether to create a queue in the RabbitMQ data source. <ul style="list-style-type: none"> <li>• If you select <b>Yes</b>, a new queue is created and the data to be integrated is sent to the queue.</li> <li>• If you select <b>No</b>, the data to be integrated is sent to an existing queue.</li> </ul> |

| Parameter          | Description  |
|--------------------|--|
| Exchange Mode      | <p>Select a routing mode for the exchange in the RabbitMQ data source to forward messages to the queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, select the routing mode for the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, select the routing mode that is the same as that of the existing destination queue.</p> <ul style="list-style-type: none"> <li>• <b>Direct</b>: If the routing key in a message fully matches the queue, the message will be forwarded to the queue.</li> <li>• <b>Topic</b>: If the routing key in a message approximately matches the queue, the message will be forwarded to the queue.</li> <li>• <b>Fanout</b>: All messages will be forwarded to the queue.</li> <li>• <b>Headers</b>: If the Headers attribute of a message fully matches the queue, the message will be forwarded to the queue.</li> </ul> |
| Exchange Name      | <p>Enter the exchange name of the RabbitMQ data source. If <b>New Queue Creation</b> is set to <b>Yes</b>, the exchange name of the new queue is used. If <b>New Queue Creation</b> is set to <b>No</b>, configure the exchange name that is the same as that of the existing destination queue.</p>   |
| Routing Key        | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Direct</b> or <b>Topic</b>.</p> <p>RabbitMQ uses the routing key as the judgment condition. Messages that meet the condition will be forwarded to the queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, enter the routing key of the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, enter the routing key that is the same as that of the existing destination queue.</p>   |
| Message Parameters | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Headers</b>.</p> <p>RabbitMQ uses Headers as a judgment condition. Messages that meet the condition will be forwarded to a new queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, enter the headers of the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, enter the headers that are the same as those of the existing destination queue.</p>  |
| Queue Name         | <p>This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b>.</p> <p>Enter the name of a new queue.</p>   |
| Automatic Deletion | <p>This parameter specifies whether a queue will be automatically deleted if no client is connected.</p>   |

| Parameter   | Description   |
|-------------|---|
| Persistence | This parameter specifies whether messages in a queue are stored permanently.  |
| Metadata    | <p>Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"> <li>• <b>Alias:</b> user-defined metadata name.</li> <li>• <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> <li>• <b>Parsing Path:</b> full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the destination is RabbitMQ:

**Figure 5-36** RabbitMQ configuration example

The screenshot shows a configuration form for a RabbitMQ queue. It includes the following sections:

- New Queue Creation:** Radio buttons for 'Yes' and 'No'.
- Exchange Mode:** A dropdown menu set to 'Direct'.
- Exchange Name:** A text input field containing 'direct'.
- Routing Key:** A text input field containing 'roma'.
- Automatic Deletion:** Radio buttons for 'Yes' and 'No'.
- Persistence:** Radio buttons for 'Yes' and 'No'.
- Metadata:** A table with columns: Alias, Type, Parsing Path, and Operation.

| Alias | Type              | Parsing Path | Operation   |
|-------|-------------------|--------------|-------------|
| c     | Integer           | a.b.c        | Edit Delete |
| d     | String yyyy-MM-dd | a.b.d        | Edit Delete |

An 'Add' button is located at the bottom left of the metadata table.

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## SQL Server

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select SQL Server as the data source type at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-63** SQL Server destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the SQL Server data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>SQL Server</b> .  |
| Data Source Name        | Select the SQL Server data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table                   | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |

| Parameter           | Description   |
|---------------------|---|
| Batch Number Format | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique. |
| Clear Table         | When this function is enabled, the destination table is cleared before tasks are scheduled.                       |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## GaussDB(for MySQL)

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled** or **Real-Time**, you can select GaussDB(for MySQL) as the destination data source.

1. On the **Create Task** page, configure destination information.

**Table 5-64** GaussDB(for MySQL) destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the GaussDB(for MySQL) data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select <b>GaussDB(for MySQL)</b> .  |
| Data Source Name        | Select the GaussDB(for MySQL) data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Table                   | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field      | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |



| Parameter           | Description   |
|---------------------|---|
| Batch Number Format | Format: yyyyMMddHHmmss/UUID<br>UUID is recommended because batch numbers in yyyyMMddHHmmss format are not unique. |
| Clear Table         | When this function is enabled, the destination table is cleared before tasks are scheduled.                       |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## Custom Data Sources

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, custom data sources can be selected at the destination.

1. On the **Create Task** page, configure destination information.

**Table 5-65** Custom data source destination information

| Parameter               | Description   |
|-------------------------|---|
| Instance                | Set this parameter to the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured.   |
| Integration Application | Select the integration application to which the custom data source belongs. Ensure that the integration application has been configured in <a href="#">Connecting to Data Sources</a> .   |
| Data Source Type        | Select a custom data source type.   |
| Data Source Name        | Select the custom data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Metadata                | Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> |

| Parameter | Description  |
|-----------|--|
|           | In addition to the preceding parameters, different custom data sources define different writer parameters. Set the parameters based on the original definition specifications of the connector. You can locate the connector used by the custom data source on the <b>Assets</b> page of the ROMA Connect console and view the writer parameter definition of the connector. |

The following figure shows an example of configuring a custom data source for sending emails. The destination is the custom data source. The **receiver** and **title** parameters are the destination parameters defined in the connector. **id**, **name**, and **info** are the data fields to be written to the custom data source.

**Figure 5-37** Custom data source configuration example

| * receiver | <input type="text" value="test@example.com"/>   |         |      |    |         |      |        |      |        |
|------------|---|---------|------|----|---------|------|--------|------|--------|
| * title    | <input type="text" value="User information"/>   |         |      |    |         |      |        |      |        |
| * Metadata | <table border="1"> <thead> <tr> <th>Alias ?</th> <th>Type</th> </tr> </thead> <tbody> <tr> <td>id</td> <td>Integer</td> </tr> <tr> <td>name</td> <td>String</td> </tr> <tr> <td>info</td> <td>String</td> </tr> </tbody> </table> | Alias ? | Type | id | Integer | name | String | info | String |
| Alias ?    | Type  |         |      |    |         |      |        |      |        |
| id         | Integer   |         |      |    |         |      |        |      |        |
| name       | String  |         |      |    |         |      |        |      |        |
| info       | String  |         |      |    |         |      |        |      |        |

2. After configuring the destination information, proceed with [Configuring a Data Mapping Rule](#).

## 5.3.4 Configuring a Data Mapping Rule

### Overview

This topic describes how to configure mapping information for a data integration task. Based on mappings between source data fields and destination data fields, ROMA Connect converts the obtained source data and writes it to the destination.

### Configuring Mapping Information

1. On the **Create Task** page, configure mapping information in auto or manual mode.

#### NOTE

- Do not use the keywords of the corresponding database as the field names at the source and destination. Otherwise, the task may become abnormal.
- If MRS Hive is used as the data source type at the destination and you need to configure partition field writing, see [Mapping Configuration of MRS Hive Partition Fields](#).

#### – **Auto Mapping Configuration**

If metadata is defined at both the source and destination, you can use the automatic mapping mode to configure mapping information.

Click **Automatic Mapping**. A mapping rule between the source and destination data fields is automatically created. If the fields are inconsistent, click **Edit** to change them or click **Add Mapping** below.

**Figure 5-38** Automatic mapping

| Destination Field | Destination FL... | Length | Source Field/Constant | Source Field Type | Length | Operation                                     |
|-------------------|-------------------|--------|-----------------------|-------------------|--------|---|
| col01             | long              | --     | col01                 | integer           | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |
| col02             | string            | --     | col02                 | string            | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |
| col03             | string            | --     | col03                 | string            | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |

[+ Add Mapping](#)

#### – Manual Mapping Configuration

You can manually add mapping rules between source data fields and destination data fields. This method applies to the integration scenario of all data types. You can configure the mapping rule by entering a key-value pair or entering a value in the text box.

- Key-Value Pair Mode

By default, the key-value pair input mode is used. Click **Add Field Mapping** to add mapping rules from source data fields to destination data fields one by one.

- Text Mode

Click **Text Mode** and enter the mapping rule script in the text box. The format is as follows:

```
[[
  {
    "sourceKey": "a1",
    "targetKey": "b1"
  },
  {
    "sourceKey": "a2",
    "targetKey": "b2"
  }
]]
```

**sourceKey** indicates a source data field, and **targetKey** indicates a destination data field. In the preceding example, source field **a1** is mapped to destination field **b1**, and source field **a2** is mapped to destination field **b2**.

2. After the mapping information is configured, if you need to configure the abnormal data storage and post-integration operation, go to [\(Optional\) Configuring Fault Information Storage](#) and [\(Optional\) Configuring the Post-Integration Operation](#). Otherwise, click **Create** to complete the data integration task configuration.

## Mapping Configuration of MRS Hive Partition Fields

When the data source type is set to MRS Hive at the destination, partition fields can be written. You can configure the partition fields based on the site requirements.

**Source Field** corresponding to the partition field must be manually entered. The specific requirements are as follows:

**Format:** {Partition field source field}.format({Character string parsing format}, {Partition field parsing format}","{year|month|day|hour|minute|second},{offset})

- If {Partition field source field} is of the String type, {String parsing format} must be specified.
- If {Partition field source field} is of the Timestamp type, {String parsing format} can be left blank.
- If {Partition field source field} is empty, the time when data is written to the destination is used as the partition field.

**Figure 5-39** Partition field mapping of MRS Hive

ⓘ Ensure that the field type of the source matches that of the destination. Otherwise, an error may occur during type conversion.

| Destination Field | Source Field                                    | Operation                                     |
|-------------------|---|---|
| id                | id  | <a href="#">Edit</a>   <a href="#">Delete</a> |
| name              | name  | <a href="#">Edit</a>   <a href="#">Delete</a> |
| yyyymm            | createtime.format("ddMMyyyy","yyyyMM",hour,1) ⓘ | <a href="#">Edit</a>   <a href="#">Delete</a> |

⊕ Add Mapping

For example, if the partition field on the destination is yyyymm and the **createtime** field on the source is used as the source field of the partition field, the time format of the **createtime** field is ddMMyyyy (day, month, and year), and the partition field on the destination is yyyyMM (year and month). If the value of the partition field is one hour later than the value of the source field, the source field name of the partition field yyyymm in the mapping information is as follows:

- If the value of **createtime** is of the String type, set this parameter to createtime.format("ddMMyyyy", "yyyyMM", hour, 1).
- If the value of **createtime** is of the Timestamp type, set this parameter to createtime.format("", "yyyyMM", hour, 1).
- If the time when data is written to the destination is used as the partition field, set this parameter to .format("", "yyyyMM", hour, 1).

### 5.3.5 (Optional) Configuring Fault Information Storage

#### Overview

This section describes how to configure data storage for abnormal data integration tasks. You can configure data storage only if the data source type is DB2, DWS, MySQL, GaussDB(for MySQL), Oracle, PostgreSQL/openGauss, SQL Server, or Gauss100 at the destination.

During each task execution, if some data at the source meets integration conditions but cannot be integrated to the destination due to network jitter or abnormal data conversion, ROMA Connect stores the data to the OBS bucket as text files.

**NOTICE**

Expand the abnormal data storage module if you need to configure it ([Table 5-66](#) describes its parameters). Otherwise, collapse the module to prevent task saving exceptions.

## Procedure

Before configuring fault information storage, ensure that you have connected the OBS data source to ROMA Connect. For details, see [Connecting to an OBS Data Source](#).

1. On the **Create Task** page, configure parameters for fault information storage.

**Table 5-66** Parameters for fault information storage

| Parameter               | Description   |
|-------------------------|---|
| Integration Application | Select the integration application to which the OBS data source belongs.  |
| Data Source Type        | This parameter can only be set to <b>OBS</b> .  |
| Name                    | Select the OBS data source that you configured.   |
| Path                    | Enter the object name of the OBS data source where abnormal data is to be stored. The value of <b>Path</b> cannot end with a slash (/). |

2. If you need to configure the post-integration operation, go to [\(Optional\) Configuring the Post-Integration Operation](#). Otherwise, click **Create** to complete the data integration task configuration.

## 5.3.6 (Optional) Configuring the Post-Integration Operation

### Overview

After a scheduled data integration task is successfully executed, messages can be sent to Kafka or MRS Kafka based on the configuration. In this way, a third-party platform (for example, DataArts Studio) can subscribe to Kafka or MRS Kafka to obtain messages and determine whether to perform data extraction.

This section describes how to configure message sending after a task is executed. If message sending is not required, you do not need to configure message sending.

### Prerequisites

- The Kafka and MRS Kafka data sources have been created under the corresponding application. For details, see [Connecting to a Kafka Data Source](#) and [Connecting to an MRS Kafka Data Source](#).

- A topic has been created and granted with the publish and subscribe permissions. For details about how to create a topic and grant permissions, see [Message Integration Guide](#).

## Procedure

1. On the **Create Task** page, configure post-integration operation information.

**Table 5-67** Parameters for configuring the post-integration operation

| Parameter              | Description   |
|------------------------|---|
| Action                 | <p>Determine whether to send a message after the task is executed.</p> <p><b>Do no send messages:</b> No message is sent after the task is successfully executed.</p> <p><b>Send messages to Kafka:</b> A message will be sent to Kafka after the task is successfully executed.</p> <p><b>Send messages to MRS Kafka:</b> A message will be sent to MRS Kafka after the task is successfully executed.</p> <p>FDI task: After an FDI task is complete, a maximum of five common scheduled tasks can be automatically or manually scheduled at the same time.</p> |
| Application            | Select the integration application.   |
| Data Source            | <p>Select the data source type that is the same as that selected for <b>Action</b>.</p> <p>For example, if <b>Send messages to Kafka</b> is selected for <b>Action</b>, select <b>Kafka</b> for <b>Data Source</b>.</p>   |
| Topic                  | Select the topic to which the message is to be sent.  |
| Custom Message Content | <p>Customize the content of the message to be sent by using <code>\${}</code> to reference variables. The supported variables are as follows:</p> <p>taskName: task name.</p> <p>dataCount: data volume.</p> <p>targetTable: destination table.</p> <p>dataSize: data size.</p> <p>For example, the content <code>\${taskName}</code> indicates that only the task name is sent.</p>  |

| Parameter | Description  |
|-----------|--|
| FDI Task  | Mandatory when <b>Action</b> is set to <b>FDI task</b> .<br>Select the next FDI task to be executed. If the selected FDI task is running when the current task is complete, the scheduling is skipped. |

2. Click **Create**.

## 5.4 Creating a Composite Data Integration Task

### 5.4.1 Configuring Oracle CDC (LogMiner)

#### Overview

Change Data Capture (CDC) enables ROMA Connect to synchronize data sources in real time and synchronously delete data tables.

ROMA Connect supports two CDC modes: XStream and LogMiner. This section describes how to enable the CDC function in LogMine mode for the Oracle database. In addition, the CDC configuration varies according to the Oracle database type (CDB or non-CDB).

#### NOTE

Currently, the CDC task supports only collection of physical database tables and does not support views.

#### Prerequisites

- If the Oracle database is deployed in primary/secondary mode, the secondary Oracle database is not used.
- Fields of the binary type, such as BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB, cannot be collected.
- If a table contains fields of the SDO\_GEOMETRY type, the Oracle database does not generate redo logs when data changes. Therefore, composite tasks cannot be used for data collection.

#### Configuration When the Oracle Database Is Not a CDB Database

It is recommended that the database administrator configure the CDC function.

1. Enable log archiving.
  - a. Run the following command to connect to the database as user **sys**:  
In practice, you can connect to the database in multiple modes. The following uses the command line mode as an example.

```
sqlplus /nolog  
CONNECT sys/password@host:port AS SYSDBA;
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on site requirements.
- b. Run the following command to check whether the log archiving function is enabled:
- ```
archive log list;
```
- If the message "Database log mode: No Archive Mode" is displayed, log archiving is disabled. Go to the next step.
  - If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled. Go to [1.f](#).
- c. Run the following command to set archive log parameters:
- ```
alter system set db_recovery_file_dest_size = 100G;  
alter system set db_recovery_file_dest = '/opt/oracle/oradata/recovery_area' scope=spfile;
```
- Among them:
- **100G** indicates the size of the log file storage space. Set this parameter based on site requirements.
  - **/opt/oracle/oradata/recovery\_area** indicates the log storage path. Set this parameter based on site requirements. Ensure that the path has been created in advance.
- d. Run the following command to enable log archiving function:

---

**NOTICE**

- Enabling the log archiving function requires database restart, which will interrupt services. Exercise caution when performing this operation.
- Archived logs occupy a large amount of disk space. If the disk space is full, services are affected. Therefore, you need to periodically delete expired archive logs.

---

```
shutdown immediate;  
startup mount;  
alter database archivelog;  
alter database open;
```

- e. Run the following command to check whether the log archiving function is enabled:
- ```
archive log list;
```
- If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled.
- f. Run the following command to exit the database:
- ```
exit;
```
2. Install the LogMiner tool.



- a. Run the following command to connect to the database instance as user

**sys:**

```
sqlplus sys/password@host:port/SID as sysdba
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
- **host** indicates the IP address of the server where the database instance is located. Set this parameter based on site requirements.
- **port** indicates the port used by the database instance. Set this parameter based on site requirements.
- **SID** indicates the name of the instance where the data to be synchronized is located. Set this parameter based on site requirements.

- b. Run the following command to check whether LogMiner is installed successfully:

```
desc DBMS_LOGMNR  
desc DBMS_LOGMNR_D
```

- If no information is displayed, LogMiner is not installed. Go to the next step.
- If information is displayed, LogMiner has been installed. Go to [3](#).

- c. Run the following commands to install LogMiner:

```
@$ORACLE_HOME/rdbms/admin/dbmslm.sql  
@$ORACLE_HOME/rdbms/admin/dbmslmd.sql
```

3. Create a LogMiner user and grant permissions to the user.

- a. Run the following commands to create a LogMiner user role and configure permissions for the role:

```
create role roma_logminer_privs;  
grant create session,  
execute_catalog_role,  
select any transaction,  
flashback any table,  
select any table,  
lock any table,  
select any dictionary to roma_logminer_privs;  
grant select on SYSTEM.LOGMNR_COL$ to roma_logminer_privs;  
grant select on SYSTEM.LOGMNR_OBJ$ to roma_logminer_privs;  
grant select on SYSTEM.LOGMNR_USER$ to roma_logminer_privs;  
grant select on SYSTEM.LOGMNR_UID$ to roma_logminer_privs;  
grant select on V_$DATABASE to roma_logminer_privs;  
grant select_catalog_role to roma_logminer_privs;  
grant LOGMINING to roma_logminer_privs;
```

Among them:

- **roma\_logminer\_privs** indicates the role name of the LogMiner user. Set this parameter based on site requirements.
- **grant LOGMINING to roma\_logminer\_privs;** can be added only if the Oracle version is 12c. Otherwise, delete this row.

- b. Run the following command to create user **LogMiner**:

```
create user roma_logminer identified by password default tablespace users;  
grant roma_logminer_privs to roma_logminer;  
alter user roma_logminer quota unlimited on users;
```

Among them:

- **roma\_logminer** indicates the name of the LogMiner user. Set this parameter based on site requirements.
  - **password** indicates the password of the LogMiner user. Set it based on site requirements.
  - **roma\_logminer\_privs** indicates the role of the LogMiner user, which is created in [3.a](#).
- c. Run the following command to modify log record parameters:  
`alter database add supplemental log data (all) columns;`
- d. Run the following command to disconnect the database:  
`exit;`

## Configuration When the Oracle Database Is a CDB Database

It is recommended that the database administrator configure the CDC function.

1. Enable log archiving.
  - a. Run the following command to connect to the database as user **sys**:  
In practice, you can connect to the database in multiple modes. The following uses the command line mode as an example.  

```
sqlplus /nolog  
CONNECT sys/password@host:port AS SYSDBA;
```

Among them:

    - **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
    - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on site requirements.
    - **port** indicates the port used by the database instance. Set this parameter based on site requirements.
  - b. Run the following command to check whether the log archiving function is enabled:  
`archive log list;`
    - If the message "Database log mode: No Archive Mode" is displayed, log archiving is disabled. Go to the next step.
    - If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled. Go to [1.f](#).
  - c. Run the following command to set archive log parameters:  

```
alter system set db_recovery_file_dest_size = 100G;  
alter system set db_recovery_file_dest = '/opt/oracle/oradata/recovery_area' scope=spfile;
```

Among them:

    - **100G** indicates the size of the log file storage space. Set this parameter based on site requirements.
    - **/opt/oracle/oradata/recovery\_area** indicates the log storage path. Set this parameter based on site requirements. Ensure that the path has been created in advance.

- d. Run the following command to enable log archiving function:

---

**NOTICE**

- Enabling the log archiving function requires database restart, which will interrupt services. Exercise caution when performing this operation.
  - Archived logs occupy a large amount of disk space. If the disk space is full, services are affected. Therefore, you need to periodically delete expired archive logs.
- 

```
shutdown immediate;  
startup mount;  
alter database archivelog;  
alter database open;
```

- e. Run the following command to check whether the log archiving function is enabled:

```
archive log list;
```

If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled.

- f. Run the following command to disconnect the database:

```
exit;
```

2. Install the LogMiner tool.

- a. Run the following command to connect to the database instance as user **sys**:

```
sqlplus sys/password@host:port/SID as sysdba
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
- **host** indicates the IP address of the server where the database instance is located. Set this parameter based on site requirements.
- **port** indicates the port used by the database instance. Set this parameter based on site requirements.
- **SID** indicates the name of the instance where the data to be synchronized is located. Set this parameter based on site requirements.

- b. Run the following command to check whether LogMiner is installed successfully:

```
desc DBMS_LOGMNR  
desc DBMS_LOGMNR_D
```

- If no information is displayed, LogMiner is not installed. Go to the next step.
- If information is displayed, LogMiner has been installed. Go to [3](#).

- c. Run the following commands to install LogMiner:

```
@$ORACLE_HOME/rdbms/admin/dbmslm.sql  
@$ORACLE_HOME/rdbms/admin/dbmslmd.sql
```

### 3. Create a LogMiner user and grant permissions to the user.

#### a. Run the following commands to create a LogMiner user role and configure permissions for the role:

```
create role c##roma_logminer_privs container=all;
grant create session,
execute_catalog_role,
select any transaction,
flashback any table,
select any table,
lock any table,
logmining,
set container,
select any dictionary to c##roma_logminer_privs container=all;
grant select on SYSTEM.LOGMNR_COL$ to c##roma_logminer_privs container=all;
grant select on SYSTEM.LOGMNR_OBJ$ to c##roma_logminer_privs container=all;
grant select on SYSTEM.LOGMNR_USER$ to c##roma_logminer_privs container=all;
grant select on SYSTEM.LOGMNR_UID$ to c##roma_logminer_privs container=all;
grant select on V_$DATABASE to c##roma_logminer_privs container=all;
grant select_catalog_role to c##roma_logminer_privs container=all;
```

**c##roma\_logminer\_privs** indicates the role name of the LogMiner user. Set this parameter based on site requirements.

#### b. Run the following command to create user **LogMiner**:

```
create user c##roma_logminer identified by password default tablespace users container=all;
grant c##roma_logminer_privs to c##roma_logminer container=all;
alter user c##roma_logminer quota unlimited on users container=all;
```

Among them:

- **c##roma\_logminer** indicates the name of the LogMiner user. Set this parameter based on site requirements.
  - **password** indicates the password of the LogMiner user. Set it based on site requirements.
  - **c##roma\_logminer\_privs** indicates the role of the LogMiner user, which is created in [3.a](#).
- c. Run the following command to modify log record parameters:
- ```
alter database add supplemental log data (all) columns;
```
- d. Run the following command to disconnect the database:
- ```
exit;
```

## 5.4.2 Configuring Oracle CDC (XStream)

### Overview

Change Data Capture (CDC) enables ROMA Connect to synchronize data sources in real time and synchronously delete data tables.

ROMA Connect supports two CDC modes: XStream and LogMiner. This section describes how to enable the CDC function in XStream mode for the Oracle database. In addition, the CDC configuration varies according to the Oracle database type (CDB or non-CDB).

### Prerequisites

- The Oracle database must have the OGG license (the OGG does not need to be installed).

- If the Oracle database is deployed in primary/secondary mode, the secondary Oracle database is not used.
- Fields of the binary type, such as BINARY, VARBINARY, TINYBLOB, BLOB, MEDIUMBLOB, and LOB, cannot be collected.

## Configuration When the Oracle Database Is Not a CDB Database

It is recommended that the database administrator configure the CDC function.

### 1. Enable log archiving and XStream.

- a. Run the following command to connect to the database as user **sys**:  
In practice, you can connect to the database in multiple modes. The following uses the command line mode as an example.

```
sqlplus /nolog  
CONNECT sys/password@host:port AS SYSDBA;
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
- b. Run the following command to enable Xstream:  

```
alter system set enable_goldengate_replication=true;
```
  - c. Run the following command to check whether the log archiving function is enabled:  

```
archive log list;
```

    - If the message "Database log mode: No Archive Mode" is displayed, log archiving is disabled. Go to the next step.
    - If the message "Database log mode: No Archive Mode" is displayed, log archiving is enabled. Go to [1.g](#).
  - d. Run the following command to set archive log parameters:  

```
alter system set db_recovery_file_dest_size = 100G;  
alter system set db_recovery_file_dest = '/opt/oracle/oradata/recovery_area' scope=spfile;
```

Among them:

    - **100G** indicates the size of the log file storage space. Set this parameter based on the site requirements.
    - **/opt/oracle/oradata/recovery\_area** indicates the log storage path. Set this parameter based on the site requirements. Ensure that the path has been created in advance.
  - e. Run the following command to enable log archiving function:

**NOTICE**

- Enabling the log archiving function requires database restart, which will interrupt services. Exercise caution when performing this operation.
- Archived logs occupy a large amount of disk space. If the disk space is full, services are affected. Therefore, you need to periodically delete expired archive logs.

```
shutdown immediate;  
startup mount;  
alter database archivelog;  
alter database open;
```

- f. Run the following command to check whether the log archiving function is enabled:

```
archive log list;
```

If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled.

- g. Run the following command to disconnect the database:

```
exit;
```

2. Create an XStream user and grant permissions to the user.

- a. Run the following command to connect to the database instance as user **sys**:

```
sqlplus sys/password@host:port/SID as sysdba
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
- **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
- **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
- **SID** indicates the name of the instance where the data to be synchronized is located. Set this parameter based on the site requirements.

- b. Run the following commands to create an XStream administrator and configure permissions for the administrator:

```
CREATE TABLESPACE xstream_admin_tbs DATAFILE '/opt/oracle/oradata/orcl/  
xstream_admin_tbs.dbf' SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;  
CREATE USER xstrmadmin IDENTIFIED BY password DEFAULT TABLESPACE xstream_admin_tbs  
QUOTA UNLIMITED ON xstream_admin_tbs;  
GRANT CREATE SESSION TO xstrmadmin;  
BEGIN  
  DBMS_XSTREAM_AUTH.GRANT_ADMIN_PRIVILEGE(  
    grantee      => 'xstrmadmin',  
    privilege_type => 'CAPTURE',  
    grant_select_privileges => TRUE,  
    container    => 'ALL'  
  );  
END;
```

Among them:

- **xstream\_admin\_tbs** indicates the tablespace name of the XStream administrator. Set it based on site requirements.
  - **/opt/oracle/oradata/orcl/xstream\_admin\_tbs.dbf** indicates the tablespace file of the XStream administrator. Set this parameter based on site requirements.
  - **xstrmadmin** indicates the username of the XStream administrator. Set it based on site requirements.
  - **password** indicates the password of the XStream administrator. Set it based on site requirements.
  - **container => 'ALL'** This row is added only if the Oracle version is 12c or later. Otherwise, delete this row.
- c. Run the following commands to create a user for connect ROMA Connect to the database and configure permissions for the user:
- ```
CREATE TABLESPACE xstream_tbs DATAFILE '/opt/oracle/oradata/orcl/xstream_tbs.dbf' SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;  
CREATE USER xstrm IDENTIFIED BY password DEFAULT TABLESPACE xstream_tbs QUOTA UNLIMITED ON xstream_tbs;  
GRANT CREATE SESSION TO xstrm;  
GRANT SELECT ON V_$DATABASE to xstrm;  
GRANT FLASHBACK ANY TABLE TO xstrm;  
GRANT SELECT ANY TABLE to xstrm;  
GRANT LOCK ANY TABLE TO xstrm;  
grant select_catalog_role to xstrm;
```
- Among them:
- **xstream\_tbs** indicates the tablespace name of the ROMA Connect connection user. Set it based on site requirements.
  - **/opt/oracle/oradata/orcl/xstream\_tbs.dbf** indicates the tablespace file of the ROMA Connect connection user. Set it based on site requirements.
  - **xstrm** indicates the name of the ROMA Connect connection user. Set it based on site requirements.
  - **password** indicates the password of the ROMA Connect connection user. Set it based on site requirements.
- d. Run the following command to modify log record parameters:
- ```
alter database add supplemental log data (all) columns;
```
- e. Run the following command to disconnect the database:
- ```
exit;
```
3. Create an XStream outbound server.

 **NOTE**

An XStream outbound server can be used only in one integration task. If the database needs to be used in multiple integration tasks, you need to create multiple XStream outbound servers.

- a. Run the following command to connect to the database instance as user **xstrmadmin**:
- ```
sqlplus xstrmadmin/password@host:port/SID
```
- Among them:

- **xstrmadmin** indicates the username of the XStream administrator, which is created in [2.b](#).
- **password** indicates the password of the XStream administrator, which is configured in [2.b](#).
- **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
- **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
- **SID** indicates the name of the instance where the data to be synchronized is located. Set this parameter based on the site requirements.

- b. Run the following command to create an XStream outbound server:

```
DECLARE
  tables DBMS_UTILITY.UNCL_ARRAY;
  schemas DBMS_UTILITY.UNCL_ARRAY;
BEGIN
  tables(1) := NULL;
  schemas(1) := 'ROMA';
  DBMS_XSTREAM_ADM.CREATE_OUTBOUND(
    server_name => 'dbzxout',
    table_names => tables,
    schema_names => schemas
  );
END;
```

Among them:

- **ROMA** indicates the schemas where the data table to be synchronized is located, that is, the schemas from which the CDC needs to capture data. Set this parameter based on site requirements.
- **dbzxout** indicates the outbound server name, which must be unique. Set this parameter based on site requirements.

- c. Run the following command to disconnect the database:

```
exit;
```

- d. Run the following command to connect to the database instance as user **sys**:

```
sqlplus sys/password@host:port/SID as sysdba
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
- **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
- **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
- **SID** indicates the name of the instance where the data to be synchronized is located. Set this parameter based on the site requirements.



- e. Run the following command to allow user **xstrm** to connect to the XStream outbound server:

```
BEGIN
  DBMS_XSTREAM_ADM.ALTER_OUTBOUND(
    server_name => 'dbzxout',
    connect_user => 'xstrm'
  );
END;
```

Among them:

- **dbzxout** is the outbound server name, which is configured in [3.b](#).
  - **xstrm** is the username for connecting to ROMA Connect. It is configured in [2.c](#).
- f. Run the following command to disconnect the database:

```
exit;
```

## Configuration When the Oracle Database Is a CDB Database

It is recommended that the database administrator configure the CDC function.

1. Enable log archiving and XStream.

- a. Run the following command to connect to the database as user **sys**:  
In practice, you can connect to the database in multiple modes. The following uses the command line mode as an example.

```
sqlplus /nolog
CONNECT sys/password@host:port AS SYSDBA;
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
- b. Run the following command to enable Xstream:
- ```
alter system set enable_goldengate_replication=true;
```
- c. Run the following command to check whether the log archiving function is enabled:
- ```
archive log list;
```
- If the message "Database log mode: No Archive Mode" is displayed, log archiving is disabled. Go to the next step.
  - If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled. Go to [1.g](#).
- d. Run the following command to set archive log parameters:
- ```
alter system set db_recovery_file_dest_size = 100G;
alter system set db_recovery_file_dest = '/opt/oracle/oradata/recovery_area' scope=spfile;
```

Among them:

- **100G** indicates the size of the log file storage space. Set this parameter based on the site requirements.

- **/opt/oracle/oradata/recovery\_area** indicates the log storage path. Set this parameter based on the site requirements. Ensure that the path has been created in advance.
- e. Run the following command to enable log archiving function:

---

**NOTICE**

- Enabling the log archiving function requires database restart, which will interrupt services. Exercise caution when performing this operation.
- Archived logs occupy a large amount of disk space. If the disk space is full, services are affected. Therefore, you need to periodically delete expired archive logs.

---

```
shutdown immediate;  
startup mount;  
alter database archivelog;  
alter database open;
```

- f. Run the following command to check whether the log archiving function is enabled:

```
archive log list;
```

If the message "Database log mode: Archive Mode" is displayed, log archiving is enabled.

- g. Run the following command to disconnect the database:

```
exit;
```

2. Create an XStream user and grant permissions to the user.

- a. Run the following command to connect to the database instance as user **sys**:

```
sqlplus sys/password@host:port/SID as sysdba
```

Among them:

- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
  - **SID** indicates the name of the CDB database where the data to be synchronized is located. Set this parameter based on the site requirements.
- b. Run the following commands to create an XStream administrator and configure permissions for the administrator:

```
CREATE TABLESPACE xstream_adm_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/  
xstream_adm_tbs.dbf' SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;  
alter session set container = ORCLPDB1;  
CREATE TABLESPACE xstream_adm_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/ORCLPDB1/  
xstream_adm_tbs.dbf' SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;  
alter session set container = CDB$ROOT;
```

```
CREATE USER c##xstrmadmin IDENTIFIED BY password DEFAULT TABLESPACE
xstream_admin_tbs QUOTA UNLIMITED ON xstream_admin_tbs CONTAINER=ALL;
GRANT CREATE SESSION, SET CONTAINER TO c##xstrmadmin CONTAINER=ALL;
BEGIN
  DBMS_XSTREAM_AUTH.GRANT_ADMIN_PRIVILEGE(
    grantee      => 'c##xstrmadmin',
    privilege_type => 'CAPTURE',
    grant_select_privileges => TRUE,
    container    => 'ALL'
  );
END;
```

Among them:

- **xstream\_admin\_tbs** indicates the tablespace name of the XStream administrator. Set it based on site requirements.
  - **/opt/oracle/oradata/ORCLCDB/xstream\_admin\_tbs.dbf** indicates the tablespace file of the XStream administrator in the CDB. Set it based on site requirements.
  - **ORCLPDB1** indicates the name of the PDB database.
  - **/opt/oracle/oradata/ORCLCDB/ORCLPDB1/xstream\_admin\_tbs.dbf** indicates the tablespace file of the XStream administrator in the PDB. Set it based on site requirements.
  - **c##xstrmadmin** indicates the username of the XStream administrator. Set it based on site requirements.
  - **password** indicates the password of the XStream administrator. Set it based on site requirements.
- c. Run the following commands to create a user for connect ROMA Connect to the database and configure permissions for the user:

```
CREATE TABLESPACE xstream_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/xstream_tbs.dbf'
SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
alter session set container = ORCLPDB1;
CREATE TABLESPACE xstream_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/ORCLPDB1/
xstream_tbs.dbf' SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;
alter session set container = CDB$ROOT;
```

```
CREATE USER c##xstrm IDENTIFIED BY password DEFAULT TABLESPACE xstream_tbs QUOTA
UNLIMITED ON xstream_tbs CONTAINER=ALL;
GRANT CREATE SESSION TO c##xstrm CONTAINER=ALL;
GRANT SET CONTAINER TO c##xstrm CONTAINER=ALL;
GRANT SELECT ON V_$DATABASE to c##xstrm CONTAINER=ALL;
GRANT FLASHBACK ANY TABLE TO c##xstrm CONTAINER=ALL;
GRANT SELECT ANY TABLE to c##xstrm CONTAINER=ALL;
GRANT LOCK ANY TABLE TO c##xstrm CONTAINER=ALL;
grant select_catalog_role to c##xstrm CONTAINER=ALL;
```

Among them:

- **xstream\_tbs** indicates the tablespace name of the ROMA Connect connection user. Set it based on site requirements.
- **/opt/oracle/oradata/ORCLCDB/xstream\_tbs.dbf** indicates the tablespace file of the ROMA Connect connection user in the CDB. Set it based on site requirements.
- **ORCLPDB1** indicates the name of the PDB database.

- **/opt/oracle/oradata/ORCLCDB/ORCLPDB1/xstream\_tbs.dbf** indicates the tablespace file of the ROMA Connect connection user in the PDB. Set it based on site requirements.
  - **c##xstrm** indicates the name of the ROMA Connect connection user. Set it based on site requirements.
  - **password** indicates the password of the ROMA Connect connection user. Set it based on site requirements.
- d. Run the following command to modify log record parameters:
- ```
alter database add supplemental log data (all) columns;
```
- e. Run the following command to disconnect the database:
- ```
exit;
```
3. Create an XStream outbound server.

 **NOTE**

An XStream outbound server can be used only in one integration task. If the database needs to be used in multiple integration tasks, you need to create multiple XStream outbound servers.

- a. Run the following command to connect to the database instance as user **c##xstrmadmin**:

```
sqlplus c##xstrmadmin/password@host:port/SID
```

Among them:

- **c##xstrmadmin** indicates the username of the XStream administrator, which is created in [2.b](#).
  - **password** indicates the password of the XStream administrator, which is configured in [2.b](#).
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
  - **SID** indicates the name of the CDB database where the data to be synchronized is located. Set this parameter based on the site requirements.
- b. Run the following command to create an XStream outbound server:

```
DECLARE
  tables DBMS_UTILITY.UNCL_ARRAY;
  schemas DBMS_UTILITY.UNCL_ARRAY;
BEGIN
  tables(1) := NULL;
  schemas(1) := 'ROMA';
  DBMS_XSTREAM_ADM.CREATE_OUTBOUND(
    server_name => 'dbzxout',
    table_names => tables,
    schema_names => schemas);
END;
```

Among them:

- **ROMA** indicates the schemas where the data table to be synchronized is located, that is, the schemas from which the CDC

- needs to capture data. Set this parameter based on site requirements.
- **dbzxout** indicates the outbound server name, which must be unique. Set this parameter based on site requirements.
- c. Run the following command to disconnect the database:  
exit;
- d. Run the following command to connect to the database instance as user **sys**:  
sqlplus sys/**password**@**host:port**/**SID** as sysdba
- Among them:
- **password** indicates the password of user **sys** of the database. You can obtain the password from the database administrator.
  - **host** indicates the IP address of the server where the database instance is located. Set this parameter based on the site requirements.
  - **port** indicates the port used by the database instance. Set this parameter based on the site requirements.
  - **SID** indicates the name of the CDB database where the data to be synchronized is located. Set this parameter based on the site requirements.
- e. Run the following command to allow user **c##xstrm** to connect to the XStream outbound server:
- ```
BEGIN
  DBMS_XSTREAM_ADM.ALTER_OUTBOUND(
    server_name => 'dbzxout',
    connect_user => 'c##xstrm'
  );
END;
```
- Among them:
- **dbzxout** is the outbound server name, which is configured in [3.b](#).
  - **c##xstrm** is the username for connecting to ROMA Connect. It is configured in [2.c](#).
- f. Run the following command to disconnect the database:  
exit;

## 5.4.3 Configuring the MySQL CDC (Binlog)

### Overview

Change Data Capture (CDC) enables ROMA Connect to synchronize data with data sources and physically delete data tables in real time.

This section describes how to enable the CDC function in Binlog mode for the MySQL database.

### Prerequisites

Fields of the binary type, such as TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB, cannot be collected.

## Procedure

It is recommended that the database administrator configure the CDC function. The following uses the Linux environment as an example.

### 1. Enable Binlog.

- a. Use a command line tool to connect to the server where the MySQL database is located and run the following command to log in to the database as user **root**:

```
mysql -uroot -ppassword
```

**password** indicates the password of user **root** of the database. You can obtain the password from the database administrator.

- b. Run the following command to check whether Binlog is enabled for the MySQL database:

```
show variables like 'log_bin';
```

- If the value of **log\_bin** is **OFF**, the Binlog function is disabled. Go to the next step.
- If the value of **log\_bin** is **ON**, the Binlog function is enabled. Run the following SQL command to check whether the parameter settings meet the requirements:

```
show variables like '%binlog_format%';  
show variables like '%binlog_row_image%';
```

The value of **binlog\_format** must be **ROW**, and the value of **binlog\_row\_image** must be **FULL**. If yes, go to [2](#). If no, go to the next step.

- c. Run the following command to exit the database:

```
exit;
```

- d. Run the following command to open the configuration file and press **i** to enter the editing mode:

```
vi /etc/my.cnf
```

- e. Add the following content to the file to enable Binlog:

```
server-id = 123  
log_bin = mysql-bin  
binlog_format = row  
binlog_row_image = full  
expire_logs_days = 10  
gtid_mode = on  
enforce_gtid_consistency = on
```

Among them:

- The value of **server-id** must be an integer greater than 1. Set this parameter based on site requirements. The value of **Server Id** set during data integration task creation must be different from the value of this parameter.
  - **expire\_logs\_days** indicates the retention period of Binlog files. Binlog files that have been retained for more than two days will be automatically deleted.
  - **gtid\_mode = on** and **enforce\_gtid\_consistency = on** are added only when the MySQL version is 5.6.5 or later. Otherwise, delete the two rows.
- f. Press **Esc** to exit the input mode, enter **:wq**, and press **Enter** to save the settings and exit.

- g. Run the following command to restart the MySQL database:  

```
service mysqld restart
```
        - h. Log in to the database as user **root** and run the following command to check whether the value of the **log\_bin** variable is **ON**, that is, whether the Binlog function is enabled:  

```
show variables like 'log_bin';
```
        - i. (Optional) When creating schema mapping for a MySQL CDC task, add **binlog\_rows\_query\_log\_events = 1** to the MySQL database configuration file in **e**.
      2. Run the following commands in the database to create a user for connecting ROMA Connect to the database and configure permissions for the user:  

```
CREATE USER 'roma'@'%' IDENTIFIED BY 'password';  
GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'roma'@'%';
```

Among them:

        - **roma** indicates the name of the ROMA Connect connection user. Set it based on site requirements.
        - **password** indicates the password of the ROMA Connect connection user. Set it based on site requirements.
      3. (Optional) If the MySQL database version is 8.0, run the following command to change the password authentication mode of the database connection user:  

```
ALTER USER roma IDENTIFIED WITH mysql_native_password BY 'password';
```

Among them:

        - **roma** is the database connection username created in **2**.
        - **password** is the password of the database connection user.
      4. Run the following command to exit the database:  

```
exit;
```

## 5.4.4 Configuring SQL Server CDC

### Overview

Change Data Capture (CDC) enables ROMA Connect to synchronize data with data sources and physically delete data tables in real time.

This section describes how to enable the CDC function for the SQL Server database.

### Prerequisites

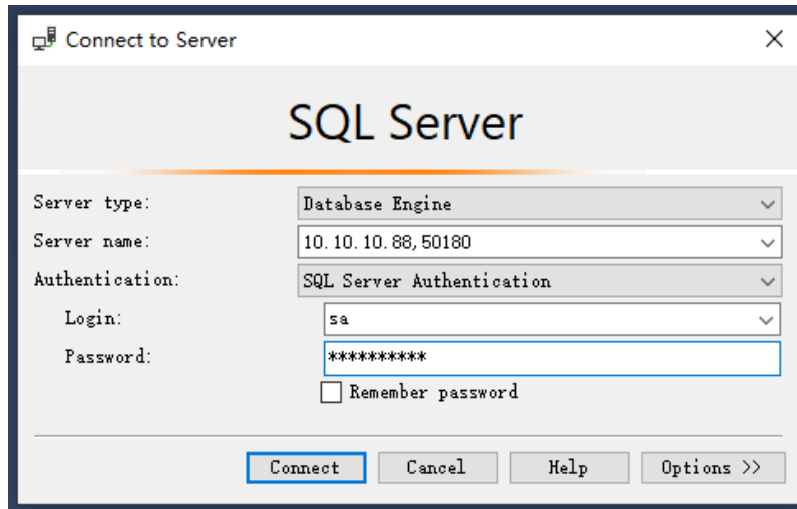
- Fields of the binary type, such as BINARY, VARBINARY, and IMAGE, cannot be collected.
- You have downloaded the SQL Server Management Studio (SSMS) client. For details, see [Download SQL Server Management Studio \(SSMS\)](#).

### Procedure

It is recommended that the database administrator configure the CDC function. The following uses the Windows environment as an example.

1. Enable CDC.

- a. Use the SSMS client to connect to the server where the SQL Server database is located, and log in to the database as user **sa**.

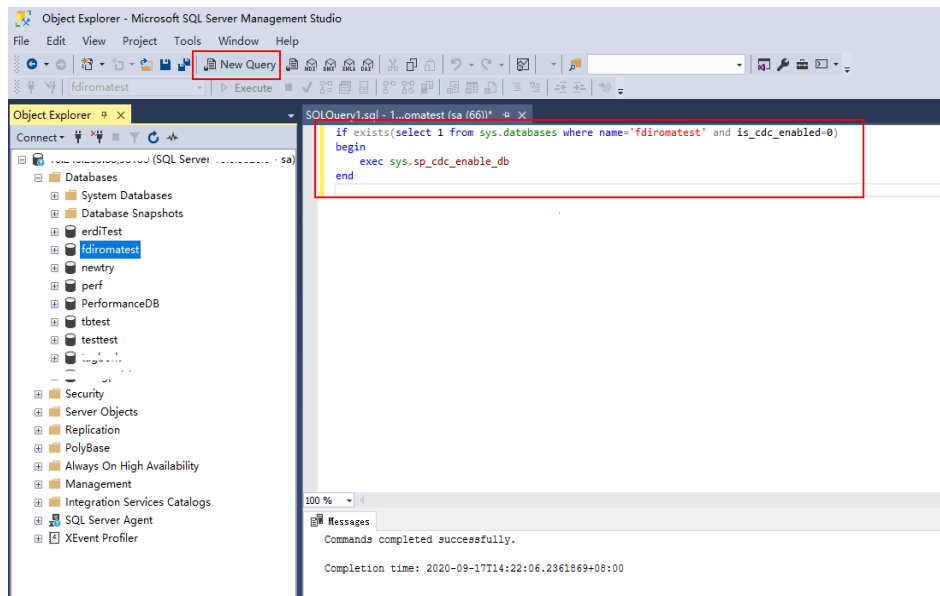


**Password** indicates the password of user **sa** of the database. You can obtain the password from the database administrator.

- b. Click **New Query** and select the database for which you want to enable CDC. Run the following command to start CDC for the SQL Server database:

```
if exists(select 1 from sys.databases where name='fdiromatest' and is_cdc_enabled=0)
begin
    exec sys.sp_cdc_enable_db
end
```

Replace *fdiromatest* with the actual database name.

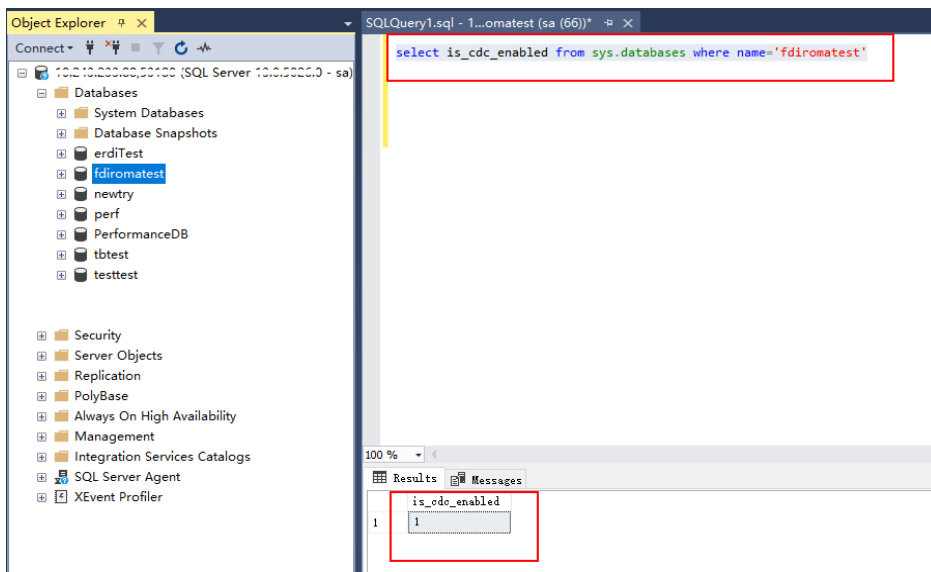


- c. Run the following command to check whether CDC is successfully enabled. If **1** is displayed, CDC is successfully enabled.

```
select is_cdc_enabled from sys.databases where name='fdiromatest'
```

Replace *fdiromatest* with the actual database name.

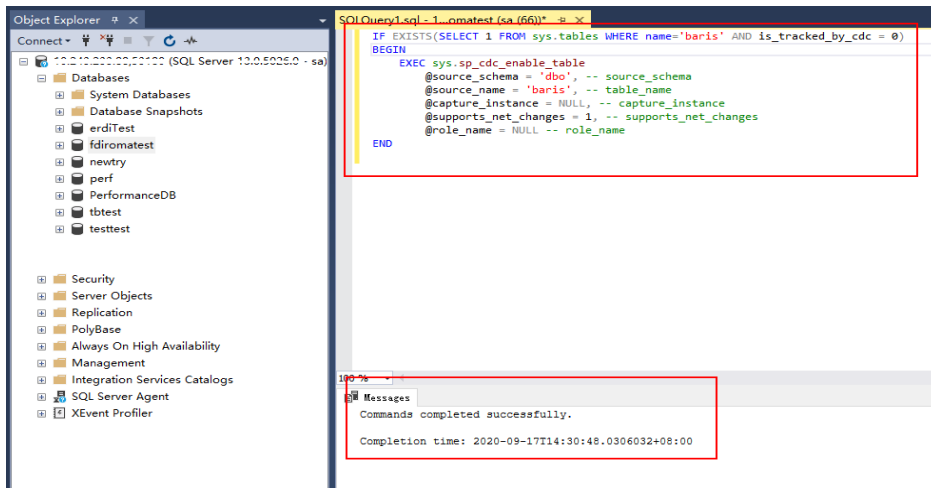




d. Enable the table-level configuration.

```
IF EXISTS(SELECT 1 FROM sys.tables WHERE name='baris' AND is_tracked_by_cdc = 0)
BEGIN
EXEC sys.sp_cdc_enable_table
    @source_schema = 'dbo', -- source_schema
    @source_name = 'baris', -- table_name
    @capture_instance = NULL, -- capture_instance
    @supports_net_changes = 1, -- supports_net_changes
    @role_name = NULL -- role_name
END
```

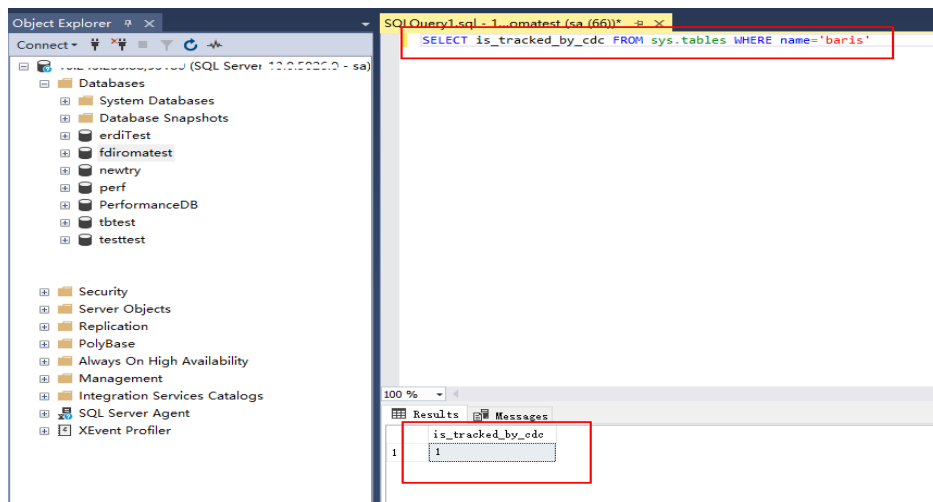
Replace *baris* with the name of the table for which CDC needs to be enabled.



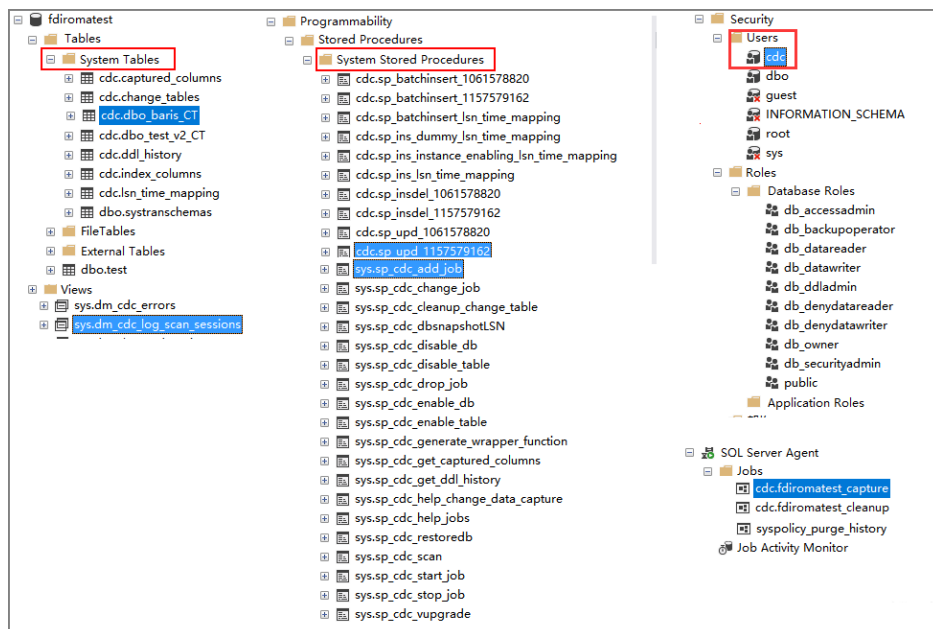
e. Run the following command to query the execution result. If the result is 1, the operation is successful.

```
SELECT is_tracked_by_cdc FROM sys.tables WHERE name='baris'
```

Replace *baris* with the name of the table for which CDC needs to be enabled.



After CDC is created, the system creates a series of system catalogs, views, stored procedures, and jobs to implement CDC functions.



2. After the CDC is configured, you need to create a user and assign permissions to the user.
  - a. Right-click **Security** and choose **New > Login** from the shortcut menu.
  - b. On the **General** page, enter the created username and password. Select **SQL Server authentication** and enter the password.
  - c. On the **Server Roles** page, select **dbcreator** and **public**.
  - d. On the **User Mapping** page, select the users and database role members that are mapped to this login.
    - i. In the **Users mapped to this login** area, select the database for which you want to configure CDC, for example, **fdiromatest**.
    - ii. In the **Database role membership for** area, select **db\_datareader**, **db\_owner**, and **public**.
  - e. Click **OK**.

## Follow-Up Operations

If the system table structure changes or the table level is adjusted, you need to enable CDC again. The configuration procedure is as follows:

1. Disable CDC and set schema and name based on the site requirements.  
EXEC sys.sp\_cdc\_disable\_table  
@source\_schema = N'dbo',  
@source\_name = 'baris',  
@capture\_instance = 'all'
2. Enable CDC again and set schema and name based on the site requirements.  
Enable the table-level configuration.

```
IF EXISTS(SELECT 1 FROM sys.tables WHERE name='baris' AND is_tracked_by_cdc = 0)
BEGIN
    EXEC sys.sp_cdc_enable_table
        @source_schema = 'dbo', -- source_schema
        @source_name = 'baris', -- table_name
        @capture_instance = NULL, -- capture_instance
        @supports_net_changes = 1, -- supports_net_changes
        @role_name = NULL -- role_name
END
```

## 5.4.5 Creating a Composite Task

### Overview

You can create a composite task if you need to continuously synchronize real-time data. This task allows FDI to implement real-time and incremental synchronization of multiple data tables from the source to destination, improving the data integration and synchronization efficiency.

The composite task supports flexible mappings of fields between data tables. For example, multiple fields in one data table at the source can be mapped to different data tables at the destination, or fields in multiple data tables at the source can be mapped to one data table at the destination.

### Prerequisites

- You have connected to data sources at the source and destination. For details, see [Connecting to Data Sources](#).  
In the data source configuration at the source, the value of **Database** must be the same as the actual database name (case-sensitive). Otherwise, data synchronization will fail.
- The CDC function has been enabled at the source. The CDC implementation modes vary depending on data source types. For details, see the following:
  - [Configuring Oracle CDC \(LogMiner\)](#) (recommended)
  - [Configuring Oracle CDC \(XStream\)](#)
  - [Configuring the MySQL CDC \(Binlog\)](#)
  - [Configuring SQL Server CDC](#)
- The retention period of CDC archiving logs in a data source at the source must be greater than the log time parsed by the integration task. Otherwise, the integration task cannot find archive logs, resulting in incremental synchronization failures. Therefore, it is not recommended that a data

integration task be stopped for a long time. It is recommended that archive logs be retained for at least two days.

- Do not perform Data Definition Language (DDL) operations on the source database during the first data synchronization.
- If a large number of composite tasks are created, the database server and FDI plug-in process will consume resources. Therefore, you are advised not to create too many composite tasks for a database.
- You can configure multiple database tables under multiple schemas in a single CDC task to implement unified collection for full or incremental data.
- During the running of a composite task, you can add a table and perform full or incremental collection on the new table after the restart.
- Synchronization is not supported for the following types of Oracle data sources at the source:

- Fields of the large text type and binary type
- A data table whose name contains lowercase letters cannot be synchronized.
- Data tables that do not have primary keys cannot be synchronized.

If a table contains only a small amount of data, collect full data once a day. Data in PostgreSQL tables can be cleared before you write to the table. If data is collected from the Oracle database but no primary key is available in the table, you can use the internal row ID of the Oracle database as the primary key. The row ID is 18 characters of digits and letters.

- Data tables or data fields whose names are reserved in the database
- Data deleted in truncate mode cannot be synchronized. Data deleted in entire table mode cannot be synchronized.

- For the MySQL data source at the source:

If the MySQL database uses the MGR cluster mode, the source data source must be directly connected to the active node instead of the route node.

If the MySQL database contains a large amount of data, the connection to the database may time out when data is synchronized for the first time. You can modify the **interactive\_timeout** and **wait\_timeout** parameters of the MySQL database to avoid this problem.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**. On the page displayed, click **Create Composite Task**.
3. On the **Create Composite Task** page, configure basic task information.

**Table 5-68** Basic task information

| Parameter                  | Description  |
|----------------------------|--|
| Name                       | After a task is created, the task name cannot be modified. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Integration Mode           | Select <b>Scheduled</b> as the mode used for data integration. <ul style="list-style-type: none"><li>• <b>Scheduled</b>: A data integration task is executed according to the schedule to integrate data from the source to the destination.</li></ul> <b>NOTE</b><br>This mode applies only to composite tasks whose data source type is MySQL, Oracle, PostgreSQL, SQL Server, or HANA. <ul style="list-style-type: none"><li>• <b>Real-Time</b>: The data integration task continuously detects updates to the data at the source and integrates updates to the destination in real time.</li></ul> The data integration mode varies depending on the data source. For details, see <a href="#">Table 5-1</a> . |
| Description                | It is recommended that you add task descriptions based on the actual task usage to differentiate tasks. The task description can be modified after being created.  |
| Tag                        | Add or select an existing tag to classify tasks for quick search. New tags are saved when you save the task and can be searched directly when you create another task.   |
| Operation Types            | Mandatory for <b>Integration Mode</b> set to <b>Real-Time</b> .<br>Select the operation types for database logs, including <b>Insert</b> , <b>Delete</b> , and <b>Update</b> . For example, if you select <b>Insert</b> and <b>Update</b> , only the logs related to data insert and update in the database are obtained.  |
| Use Quartz Cron Expression | Mandatory for <b>Integration Mode</b> set to <b>Scheduled</b> .<br>Schedule a task using a Quartz cron expression.   |
| Period                     | Mandatory for <b>Integration Mode</b> set to <b>Scheduled</b> .<br>Set the task execution interval, from minutes to months.<br>For example, if <b>Unit</b> is set to <b>Day</b> and <b>Period</b> is set to <b>1</b> , the task is executed once every day.  |

| Parameter          | Description   |
|--------------------|---|
| Expression         | <p>Mandatory for <b>Use Quartz Cron Expression</b> set to <b>Yes</b>. Configure a Quartz cron expression for task scheduling. <b>Second</b> in the expression is fixed to <b>0</b> because ROMA Connect supports only down-to-minute schedules. For details, see <a href="#">Appendix: Quartz Cron Expression Configuration</a>.</p> <p>A task needs to be executed every 15 minutes from 01:00 a.m. to 04:00 a.m. every day. In this example, the expression should be as follows:</p> <pre>0 0/15 1-4 * * ?</pre> |
| Effective Time     | <p>Mandatory for <b>Integration Mode</b> set to <b>Scheduled</b>. Start time of a task.</p>   |
| Sync Existing Data | <p>Available only after you click <b>Edit</b> of a task. This function takes effect when you add a table mapping to a composite task that has started.</p> <ul style="list-style-type: none"> <li>When enabled, the task synchronizes all the existing data of the newly added tables. After that, data is synchronized in increments.</li> <li>When disabled, the task synchronizes only data generated from the start of the task.</li> </ul>   |

4. Configure a mapping between data sources at the source and destination.

**Table 5-69** Source and destination configuration information

| Parameter   |                         | Description  |
|-------------|-------------------------|--|
| Source      | Instance Name           | Select the ROMA Connect instance that is being used.   |
|             | Integration Application | Select the integration application to which the data source at the source belongs.   |
|             | Data Source Type        | Select a data source type at the source.<br>Scheduled: MySQL, Oracle, SQL Server, PostgreSQL, HANA<br>Real-time: MySQL, Oracle, and SQL Server   |
| Destination | Instance Name           | Select the ROMA Connect instance that is being used. After the source instance is configured, the destination instance is automatically associated and does not need to be configured. |
|             | Integration Application | Select the integration application to which the data source at the destination belongs.  |

| Parameter |                  | Description   |
|-----------|------------------|---|
|           | Data Source Type | Select a data source type at the destination.<br>Scheduled: MySQL, Oracle, PostgreSQL, SQL Server, and HANA<br>Real-time: MySQL, Oracle, PostgreSQL, Kafka, and SQL Server, |

5. Configure data table mappings between the source and destination in manual or automatic mode.

 **NOTE**

- The length of a data field at the destination must be greater than or equal to that of the data field at the source. Otherwise, the synchronized data will be lost.
  - A maximum of 1000 data tables can be synchronized in a task.
  - If the data source type at the destination is **Kafka**, the table displayed on the destination is a virtual table. You only need to edit the field mappings in the table.
  - For the Oracle destination data source, if the source primary key field is empty, the record is discarded by default and no scheduling log error code is generated.
- Automatic mapping
    - i. Click **Automatic Mapping**. In the dialog box that is displayed, configure the mapping policy and range and then click **Start Mapping**. The mapping between data tables will be automatically generated.
    - ii. Click **Edit** to modify a mapping between data tables as required.
    - iii. Click **View**. In the dialog box displayed, modify the mappings between fields in the data tables as required or add new mappings.  
The length of a data field at the destination must be greater than or equal to that of the data field at the source. Otherwise, the synchronized data will be lost.
  - Manually adding a table mapping
    - i. In the **Table Mappings** area, click **Add** to manually add a table mapping.
    - ii. Set **Source Table Name** and **Destination Table Name** for the table mapping.
    - iii. Click **Map** in the **Operation** column. In the window displayed, view or edit the mapping fields or delete unnecessary fields, or click **Add**



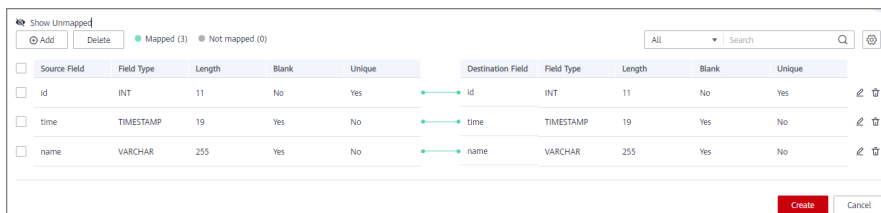
to add a field mapping. Click  in the upper right corner to set the configuration items for the field mapping:

- **Source Field:** Select a field name in the source table, for example, **ID**.
- **Destination Field:** Select the corresponding field name in the destination table, for example, **Name**.
- **Prefix:** Enter the prefix of the synchronization field.

- **Suffix:** Enter the suffix of the synchronization field.

The following is an example of configuring the prefix and suffix. For example, if the field content is **test**, the prefix is **tab1**, and the suffix is **1**, the field after synchronization is **tab1test1**.

**Figure 5-40** Configuring field mappings



Function mapping is available only when PostgreSQL is set as the destination. Click **Add Function Mapping** to map functions.

- **Mapping Function:** Select a mapping relationship.
  - **Destination Field:** Select a destination field to map, for example, **Name**.
- iv. Click **Save**.
6. Configure abnormal data storage.

**NOTE**

This configuration is available only when the data source type at the destination is MySQL, Oracle, PostgreSQL, or SQL Server. Before the configuration, connect to the OBS data source. For details, see [Connecting to an OBS Data Source](#).

During each task execution, if some data at the source meets integration conditions but cannot be integrated to the destination due to network jitter or other exceptions, ROMA Connect stores the data to the OBS bucket as text files.

**Table 5-70** Abnormal data storage information

| Parameter               | Description   |
|-------------------------|---|
| Source Data Type        | This parameter can only be set to <b>OBS</b> .  |
| Integration Application | Select the required integration application.  |
| Name                    | Select the OBS data source that you configured.   |
| Path                    | Enter the object name of the OBS data source where abnormal data is to be stored. The value of <b>Path</b> cannot end with a slash (/). |

7. Set transaction thresholds.

**NOTE**

Transaction thresholds are available only when the integration mode is set to **Real-time** and the source data type set to **Oracle**.



For Oracle sources, transactions whose size or duration exceeds the values set for the following two parameters are forcibly committed.

**Table 5-71** Setting transaction thresholds

| Parameter                  | Description      |
|----------------------------|------------------|
| Transaction Size           | Default: 100,000 |
| Transaction Duration (min) | Default: 250     |

8. Click **Create**.

 **NOTE**

In the following scenarios, click **Reset** in the **Operation** column of a composite task to reset its synchronization. Then the task starts to synchronize existing data again and later synchronize increments in real time.

- Composite tasks need to support synchronization of new data tables and data fields at the source.
- The CDC archive logs at the source are cleared. As a result, the composite task fails to be synchronized.
- The MySQL database does not use the GTID mode, and an active/standby switchover occurs. As a result, the composite task fails to be synchronized.

You can reset the task only when **Task Status** is **Stopped**.

## 5.5 Creating a Flow Data Integration Task

### 5.5.1 Configuring a Flow Task Process

#### Overview

Flow tasks can be created on the console with ease, with the following functions provided:

- A processor node can collect data from one data source to multiple destinations.
- Tasks can be configured simply by dragging and dropping nodes.
- Data conversion modes include the mapping mode and script mode. With JavaScript scripts, you can read data from the source, process the data, and then write the data to the destination.


#### Prerequisites

- The source and destination data sources have been connected to ROMA Connect. For details, see [Connecting to Data Sources](#).
- ROMA Connect has the permission to write data to the destination.
- If you need to configure data storage with abnormal synchronization, ensure that the OBS data source has been connected. For details, see [Connecting to an OBS Data Source](#).


## Procedure


1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**. On the displayed page, click **Create Flow Task**.
3. Add source nodes.
  - a. Drag **Integration Application** under **Basic Components** on the left to the orchestration area.
  - b. Click the **Integration Application** node for configuration. For details, see [Table 5-72](#). After the configuration is complete, click **OK**.

**Table 5-72** Integration application parameters

| Parameter   | Description   |
|-------------|---|
| Instance    | Select the instance of the project.   |
| Application | Select the application to which the instance belongs.<br>If no application is available, click  to create an integration application. For details, see <a href="#">Creating an Integration Application</a> . |

### NOTE

Click  in the upper left corner of the dialog box to change the name of the integration application.

- c. In the **Data Source** area on the left, select a data source type, click , and drag a data source component to the integration application as a source node.
  - d. Click the data source node and select the corresponding data source.  
If no data source is available, click **Add** to add one. For details, see [Data Source Management](#).
4. Add a destination node.
    - a. Drag **Integration Application** under **Basic Components** on the left to the orchestration area.
    - b. Add an integration application for the data source. For details, see [3](#).
  5. Add a processor node.
    - a. Drag **Task Scheduler** under **Processor** on the left to the orchestration area. Task schedulers schedule data between data sources and set the task integration mode and execution period. The following table describes the parameters.

**Table 5-73** Task scheduler parameters

| Parameter              | Description   |
|------------------------|---|
| Integration Mode       | Select <b>Scheduled</b> as the mode used for data integration. <ul style="list-style-type: none"><li>• <b>Scheduled</b>: A data integration task is executed according to the schedule to integrate data from the source to the destination.</li><li>• <b>Real-Time</b>: The data integration task continuously detects updates to the data at the source and integrates updates to the destination in real time.</li></ul> The data integration mode varies depending on the data source. For details, see <a href="#">Table 5-1</a> . |
| Parse                  | Specify whether to transparently transmit data, that is, pass data from the source to the destination without altering the data. <ul style="list-style-type: none"><li>• <b>Yes</b></li><li>• <b>No</b></li></ul>   |
| Quartz Cron Expression | Specify whether to adopt Quartz cron expressions. <ul style="list-style-type: none"><li>• See <a href="#">Using Quartz cron expressions</a> if you use Quartz cron expressions.</li><li>• See <a href="#">Not using Quartz cron expressions</a> if you do not use Quartz cron expressions.</li></ul>  |
| Execution Time         | Start time of a task.   |
| Description            | Enter the description of the task scheduler.  |

- b. Configure the task scheduler. Choose whether to use Quartz cron expressions for configuration.

▪ **Using Quartz cron expressions**

The Quartz Cron expression supports flexible schedules. For example, a task can be executed every 15 minutes from 01:00 a.m. to 04:00 a.m. every day based on the settings of the Quartz Cron expression. Such a schedule cannot be implemented through simple GUI configuration, but using the Quartz Cron expression.

```
0 0/15 1-4 * * ?
```

**Figure 5-41** Quartz cron expression parameters

The screenshot displays a configuration form for Quartz cron expressions. It includes the following elements:

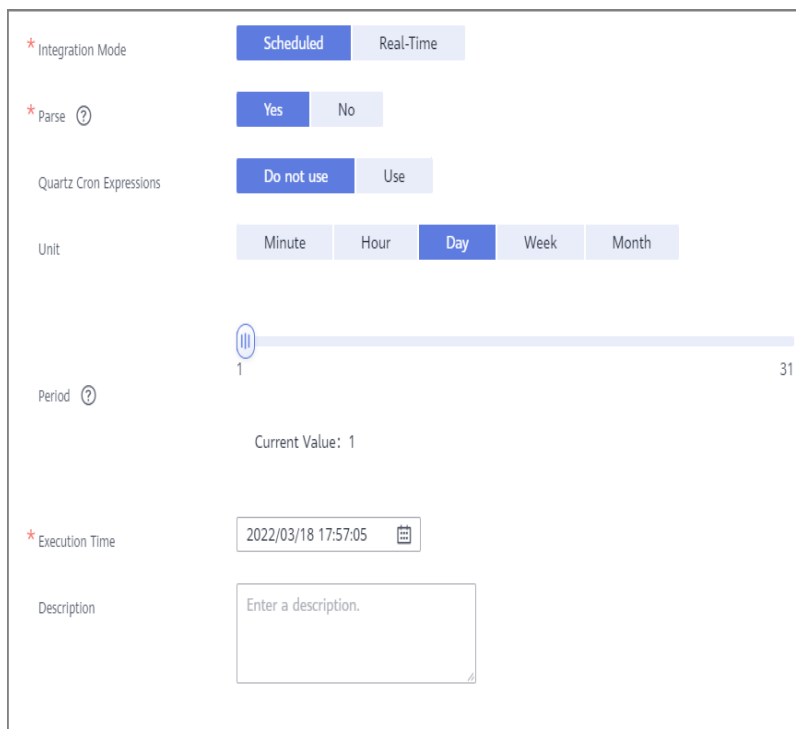
- \* Integration Mode:** Radio buttons for "Scheduled" (selected) and "Real-Time".
- \* Parse:** Radio buttons for "Yes" (selected) and "No".
- Quartz Cron Expressions:** Radio buttons for "Do not use" and "Use" (selected).
- \* pavementFDI.business.task.quart:** A cron expression field with sub-inputs for Second (0), Minute (0), Hour (0), Day (12), Month (\*), and Week (?).
- \* Execution Time:** A date and time field showing "2022/03/18 17:57:05" with a calendar icon.
- Description:** A text area with the placeholder "Enter a description."

**Table 5-74** Quartz cron expression parameters

| Parameter              | Description  |
|------------------------|--|
| Quartz Cron Expression | Select <b>Yes</b> .  |
| Quartz Cron Expression | Specify the Quartz Cron expression for the schedule. Currently, ROMA Connect supports only minute-level schedules. The second in the expression is fixed to <b>0</b> . For details, see <a href="#">Appendix: Quartz Cron Expression Configuration</a> .<br>For example, if a task needs to be executed every 15 minutes from 01:00 a.m. to 04:00 a.m. every day, the Quartz Cron expression corresponding to the schedule is as follows:<br>0 0/15 1-4 ** ? |

- **Not using Quartz cron expressions**

**Figure 5-42** GUI configuration

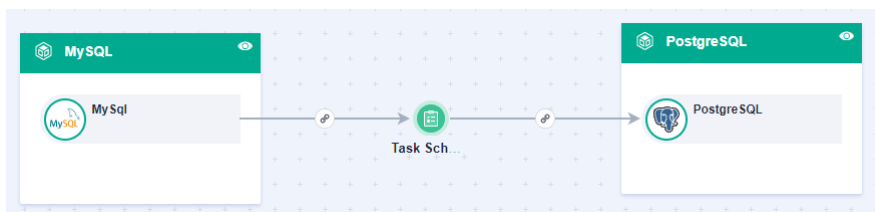



**Table 5-75** Simple GUI configuration

| Parameter              | Description  |
|------------------------|--|
| Quartz Cron Expression | Select <b>Do not use</b> .   |
| Unit                   | Set the unit of the task execution period. The value can be <b>Minute, Hour, Day, Week, or Month</b> .   |
| Period                 | Set the task execution cycle. The value range varies depending on the value of <b>Unit</b> .<br>For example, if <b>Unit</b> is set to <b>Day</b> and <b>Period</b> is set to <b>1</b> , the data integration task is executed once a day. Ensure that the scheduling period is longer than the task execution period to prevent task failures. |

- c. After the configuration is complete, click **OK**.
- 6. Configure orchestration task connections.
  - a. Connect the data source node at the source, task scheduler, and data source node at the destination in sequence.

Figure 5-43 Connecting components



- b. Click the line between the source data node and task scheduler for configuration. For details, see [Configuring Source Information](#). After the configuration is complete, click **OK**.
  - c. Click the line between the task scheduler and destination node and configure line parameters. For details, see [Configuring Destination Information](#). After the configuration is complete, click **OK**.
  - d. Click  in the upper right corner of the page and check whether there are incorrect configurations.
    - If yes, reconfigure the nodes.
    - If no, go to the next step.
7. Click **Save** in the upper right corner of the page to save the orchestration task.

## 5.5.2 Configuring Source Information

### Overview

This section describes how to configure source information for a flow task. Based on the source information, ROMA Connect integrates data, including the data source type, data format, and data range. The source information configuration varies depending on data source types.

| Data Source Types Supported by Scheduled Integration Tasks   | Data Source Types Supported by Real-Time Integration Tasks   |
|--|--|
| <ul style="list-style-type: none"> <li>• <a href="#">API</a></li> <li>• <a href="#">DB2</a></li> <li>• <a href="#">DWS</a></li> <li>• <a href="#">MySQL</a></li> <li>• <a href="#">MongoDB</a></li> <li>• <a href="#">OBS</a></li> <li>• <a href="#">Oracle</a></li> <li>• <a href="#">PostgreSQL</a></li> <li>• <a href="#">SQL Server</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">ActiveMQ</a></li> <li>• <a href="#">ArtemisMQ</a></li> <li>• <a href="#">Kafka</a></li> <li>• <a href="#">RabbitMQ</a></li> </ul> |

## API

If **Integration Mode** is set to **Scheduled**, API can be selected as the source data source.

**Table 5-76** API source information

| Parameter               | Description   |
|-------------------------|---|
| Paging                  | <p>This parameter specifies whether data is returned on multiple pages when ROMA Connect sends a request to the API data source to obtain data. Multiple data records can be returned for one API request.</p> <ul style="list-style-type: none"><li>• If <b>Paging</b> is enabled, all data that meets the conditions is displayed on multiple pages based on the fixed number of records on each page. Each time an integration task is executed, ROMA Connect sends multiple API requests to obtain all data. That is, each API request is sent to obtain data on one page.</li><li>• If <b>Paging</b> is disabled, ROMA Connect obtains all data that meets the conditions through one API request.</li></ul> |
| Page Number Field       | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter a page number field defined in the API data source, for example, <b>pageNo</b>. This parameter is carried when ROMA Connect sends an API request to the source to specify the number of the page from which data is to be obtained. The page number field must be configured in <b>Params</b> or <b>Body</b> of <b>Request Parameters</b>.</p>  |
| Value                   | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p><b>Value</b> indicates whether the page number starts from 0 or 1. Set <b>Value</b> based on the original definition of the API.</p>  |
| Page Size Field         | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter a page size field defined in the API data source, for example, <b>pageSize</b>. This parameter is carried when ROMA Connect sends an API request to the source to specify the maximum number of records on each page.</p>   |
| Records                 | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Set the number of records on each page based on the original definition of the API.</p>   |
| Maximum Number of Pages | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>This parameter specifies the maximum number of pages that can be queried in each scheduled task, for example, 10. If the number of pages exceeds the specified value, the task is stopped. The value <b>0</b> indicates that no restriction applies.</p>  |

| Parameter                 | Description   |
|---------------------------|---|
| Pagination End            | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Select the method to stop obtaining source data in pagination mode.</p> <ul style="list-style-type: none"> <li>• <b>Empty page list:</b> If no data record is returned, ROMA Connect stops obtaining source data.</li> <li>• <b>Number of records:</b> ROMA Connect compares the calculation result between the number of requested pages and the page size with the total number of records to determine whether to stop obtaining source data.</li> </ul>   |
| Pagination End Field Path | <p>This parameter is mandatory only if <b>Paging</b> is enabled.</p> <p>Enter the path of the field in an API response, which is used to determine the end of pagination. In the API response, elements in different layers are separated by periods (.). For example, if element c in the {"a":{"b":{"c":"xxx"}}} response is the pagination end field, the pagination end field path is set to <b>a.b.c</b>.</p> <ul style="list-style-type: none"> <li>• If <b>Pagination End</b> is set to <b>Empty page list</b>, set this parameter to the root path of the list field.</li> <li>• If <b>Pagination End</b> is set to <b>Number of records</b>, set this parameter to the path of the total number of records.</li> </ul> |
| Incremental Migration     | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Start Time Field          | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Enter the start time field originally defined in the API data source, for example, <b>startTime</b>. This parameter is carried when ROMA Connect sends an API request to the source, indicating that data following the specified field will be obtained.</p> <p>The start time field and end time field must be both entered in <b>Params</b> or <b>Body</b> of the <b>Request Parameters</b>.</p>   |
| End Time Field            | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Enter the end time field originally defined in the API data source, for example, <b>endTime</b>. This parameter is carried when ROMA Connect sends an API request to the source, indicating that data before the specified value will be obtained.</p>  |
| Time Zone                 | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the API data source so that ROMA Connect can identify the data timestamps.</p>   |



| Parameter                    | Description  |
|------------------------------|--|
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p> <p>Assume that <b>Start Time Field</b> is <b>startTime</b>, <b>End Time Field</b> is <b>endTime</b>, <b>Timestamp Initial Value</b> is <b>2020-11-01 12:00:00</b>, <b>Compensation Period</b> is <b>0</b>, and <b>Period Settings</b> is <b>Default</b> for incremental collection. If the first scheduling time of the task is 2020-11-01 13:00:00, the data collected for the first time is that the value of <b>startTime</b> is greater than or equal to 2020-11-01 12:00:00 and the value of <b>endTime</b> is less than or equal to 2020-11-01 13:00:00. For subsequent collection, the data collected each time is that the value of <b>startTime</b> is greater than or equal to the time when the task is successfully executed last time and the value of <b>endTime</b> is less than or equal to the time. Execution time of the current task.</p> |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms).</p>  |
| Time Format                  | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select a timestamp format, for example, <i>yyyy-MM-dd</i>.</p>   |

| Parameter                | Description  |
|--------------------------|--|
| Period Settings          | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the mode used for setting the time range for subsequent data integration after an incremental migration task is executed for the first time.</p> <ul style="list-style-type: none"> <li>• <b>Default:</b> Data generated between the previous scheduling and current scheduling is integrated. When ROMA Connect obtains data from the source, it uses the triggering time of the two tasks as the start time and end time, respectively.</li> <li>• <b>Custom:</b> The start time and end time are determined based on the configured period rules. This mode applies to common periodic tasks, for example, tasks executed once a day, a week, or a month.</li> </ul> |
| Start Time Offset (Days) | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Default</b>.</p> <p>Set the number of days before the start time of data collection.</p> <p>If data generated at the source changes in real time, such as alarm data, you can collect the data by setting this parameter.</p> <p>Start time of data collection = Data source system time – Start time offset</p>  |
| Time Interval            | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>Select the time granularity. The value must be the same as the unit configured in the <b>task schedule</b> so that the new data can be overwritten. For example, if <b>Unit</b> is set to <b>Day</b> in a task schedule, set this parameter to <b>Day</b>, indicating that data is obtained once a day.</p>  |
| Period                   | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>Select the time period for obtaining source data. For example, if the task is executed once a day, <b>Time Interval</b> is set to <b>Day</b>, and <b>Period</b> is set to <b>Previous period</b>, data of the previous day is incrementally integrated once. If <b>Period</b> is set to <b>Current period</b>, data of the current day is incrementally integrated once.</p>   |
| Right Periodic Boundary  | <p>This parameter is mandatory only if <b>Period Settings</b> is set to <b>Custom</b>.</p> <p>This parameter specifies whether the end time is included in the time range for obtaining source data.</p> <ul style="list-style-type: none"> <li>• <b>Closed interval:</b> The end time is included.</li> <li>• <b>Open interval:</b> The end time is not included.</li> </ul>  |
| Request Parameters       | <p>Construct the parameter definition of the API request, for example, the page number and page size fields must be carried in <b>Params</b> or <b>Body</b>. Set this parameter based on the definition of the API data source.</p>  |

| Parameter       | Description   |
|-----------------|---|
| Response Type   | Select the format that will be used for the response of an API request. The value can be <b>JSON</b> or <b>XML</b> . Ensure that the format is the same as the actual response format of the API.   |
| Data Root Field | This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON or XML format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .  |
| Metadata        | This parameter specifies each underlying key-value data element that is obtained from the source in JSON or XML format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the response.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON or XML format does not contain arrays:

For example, in the following JSON data (similar to XML data), the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON or XML format contains arrays:

For example, in the following JSON data (similar to XML data), the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding data in JSON or XML format that contains arrays is used as an example. The following describes the configuration when the destination is API:

- In the example of pagination configuration, **pageNo** and **pageSize** are the pagination parameters of the API and need to be added to the **Request Parameters**.

**Figure 5-44** API pagination configuration example

The screenshot shows the 'Edge' configuration window for API pagination. The configuration includes the following fields and values:

- \* Paging:
- \* Page Number Field (?):
- \* Value:
- \* Page Size Field (?):
- \* Records:
- Maximum Number of Pages (?):
- Pagination End:
- Pagination End Field Path:
- \* Incremental Migration:

Request Parameters

| Key      | Value | Operation                                     |
|----------|-------|---|
| pageNo   | 1     | <a href="#">Edit</a>   <a href="#">Delete</a> |
| pageSize | 5     | <a href="#">Edit</a>   <a href="#">Delete</a> |

- In the example of incremental migration configuration, **startTime** and **endTime** are the time parameters of the API and need to be added to the **Request Parameters**.

**Figure 5-45** API incremental migration configuration example

Request Parameters

| Key       | Value | Operation   |
|-----------|-------|-------------|
| startTime |       | Edit Delete |
| endTime   |       | Edit Delete |

- In the example of metadata configuration, **Data Root Field** is set to **a**.

**Figure 5-46** API metadata configuration example

| Alias | Type    | Parsing Path | Operation   |
|-------|---------|--------------|-------------|
| c     | Integer | b[1].c       | Edit Delete |
| d     | String  | b[1].d       | Edit Delete |

## ActiveMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, ActiveMQ can be selected as the source data source.

**Table 5-77** ActiveMQ information at the source

| Parameter        | Description   |
|------------------|---|
| Destination Type | Select the message transfer model of the ActiveMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name | Enter the name of an existing topic or queue from which data is obtained.   |
| Data Root Field  | This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .   |
| Metadata         | This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.

- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.
- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```

{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}

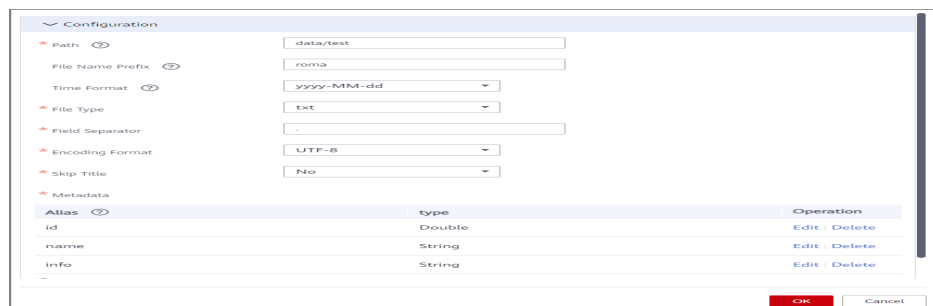
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.
- **Data Root Field** is set to **a**.  
**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.
- **Data Root Field** is set to **a.b**.  
**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is ActiveMQ:

Figure 5-47 ActiveMQ configuration example



## ArtemisMQ

[Back to Overview](#)



If **Integration Mode** is set to **Real-Time**, you can select ArtemisMQ as the data source type at the source.

1. On the **Create Task** page, configure source information.

**Table 5-78** ArtemisMQ source information

| Parameter        | Description   |
|------------------|---|
| Destination Type | Select the message transfer model of the ArtemisMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .   |
| Destination Name | Enter the name of an existing topic or queue from which data is obtained.   |
| Data Root Field  | This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .   |
| Metadata         | This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.

**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

- **Data Root Field** is set to **a**.

**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.

- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.

**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

- **Data Root Field** is set to **a**.

**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.

- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The configuration when the source is ArtemisMQ is similar to that when the source is ActiveMQ. For details, see [ActiveMQ configuration example](#).

## DB2

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, DB2 can be selected as the source data source.

**Table 5-79** DB2 source information

| Parameter             | Description   |
|-----------------------|---|
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A, B, or C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the DB2 data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize, such as the ID.</p>   |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the DB2 data source so that ROMA Connect can identify the data timestamps.</p>   |
| Timestamp Field       | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type as the timestamp, which can be used to check whether a data row meets incremental integration conditions. If the entered values of <b>Timestamp</b> field and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.</p>  |

| Parameter                    | Description  |
|------------------------------|--|
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>   |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms).</p>    |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata            | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name</b>: name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type</b>: data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path</b>: complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is DB2 is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

## DWS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, DWS can be selected as the source data source.

**Table 5-80** DWS source information

| Parameter             | Description   |
|-----------------------|---|
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the col01 and col02 columns from table01, filters data rows whose values are A, B, or C in the col02 column, and sorts the data rows in sequence based on the values in the col01 column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the DWS data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the DWS data source so that ROMA Connect can identify the data timestamps.</p>   |
| Timestamp Field       | <p>This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.</p> <p>Select a field of the DATE type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>  |

| Parameter                    | Description  |
|------------------------------|--|
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>   |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p>    |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata            | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>● <b>Field Name</b>: name of the data field whose child elements need to be collected in the table.</li><li>● <b>Type</b>: data type of the data element to be collected in the JSON field value.</li><li>● <b>Parsing Path</b>: complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is DWS is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

## Kafka

[Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, Kafka can be selected as the source data source.

**Table 5-81** Kafka information at the source

| Parameter  | Description  |
|------------|--|
| Topic Name | Select the name of the topic whose data is to be obtained. |



| Parameter       | Description   |
|-----------------|---|
| Data Root Field | This parameter specifies the path of the upper-layer common fields among all metadata in the data obtained from the source in JSON format. <b>Data Root Field</b> and <b>Parsing Path</b> in <b>Metadata</b> form a complete metadata path. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a> .   |
| Data Type       | Select the data type obtained from the Kafka data source and must be consistent with the actual data type stored in Kafka. The option is <b>JSON</b> .  |
| Offset          | Select whether to integrate the earliest message data or the latest message data.   |
| Time Zone       | Select the time zone used by the Kafka data source so that ROMA Connect can identify the data timestamps.   |
| Metadata        | This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: path of the metadata, which does not contain the data root field. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.  
**Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.
- **Data Root Field** is set to **a**.

**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b.c**, and **Parsing Path** of element d must be set to **b.d**.

- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **c**, and **Parsing Path** of element d must be set to **d**.

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    },
    {
      "c": "yy",
      "d": "yy"
    }
  ]
}
```

In this scenario, three configuration solutions are available for **Data Root Field** and **Parsing Path**:

- **Data Root Field** is not specified.

**Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

- **Data Root Field** is set to **a**.

**Parsing Path** starts from the underlying path of element a. **Parsing Path** of element c must be set to **b[i].c**, and **Parsing Path** of element d must be set to **b[i].d**.

- **Data Root Field** is set to **a.b**.

**Parsing Path** starts from the underlying path of element b. **Parsing Path** of element c must be set to **[i].c**, and **Parsing Path** of element d must be set to **[i].d**.

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is Kafka:

**Figure 5-48** Kafka configuration example

The screenshot shows a configuration form for Kafka. It includes the following fields and options:

- \* Topic Name:** A dropdown menu with 'topic-doc' selected and a refresh icon.
- Data Root Field:** A text input field containing 'a'.
- \* Data Type:** A dropdown menu with 'JSON' selected.
- Offset:** Two radio buttons, 'Earliest' (selected) and 'Latest'.
- \* Time Zone:** A dropdown menu with 'GMT+08:00' selected.
- \* Metadata:** A section with a 'Text Mode' toggle and a table.

| Alias ? | type    | Parsing Path ? | Operation                                   |
|---------|---------|----------------|---|
| c       | Integer | b.c            | <a href="#">Edit</a> <a href="#">Delete</a> |
| d       | String  | b.d            | <a href="#">Edit</a> <a href="#">Delete</a> |

At the bottom of the metadata section, there is an 'Add' button with a plus icon.

## MySQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MySQL can be selected as the source data source.

**Table 5-82** MySQL source information

| Parameter  | Description  |
|------------|--|
| Insert SQL | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |

| Parameter                    | Description   |
|------------------------------|---|
| Table                        | Mandatory when <b>Insert SQL</b> is disabled.<br>Select the data table from which data is to be obtained in the MySQL data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.  |
| Field Sorting                | Mandatory when <b>Insert SQL</b> is disabled.<br>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b> .  |
| Incremental Migration        | This parameter specifies whether only data generated in a specific period is integrated.<br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.   |
| Time Zone                    | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the MySQL data source so that ROMA Connect can identify the data timestamps.  |
| Timestamp Field              | This parameter is mandatory only if <b>Incremental Migration</b> is enabled and <b>Insert SQL</b> is disabled.<br>Select a field of the DATE type as the timestamp, which can be used to check whether a data line meets data integration conditions. If the entered values of <b>Timestamp</b> field and <b>Timestamp Initial Value</b> are incomplete, full integration is used by default.                   |
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task. |

| Parameter                | Description  |
|--------------------------|--|
| Compensation Period (ms) | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p>    |
| Filter                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata        | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

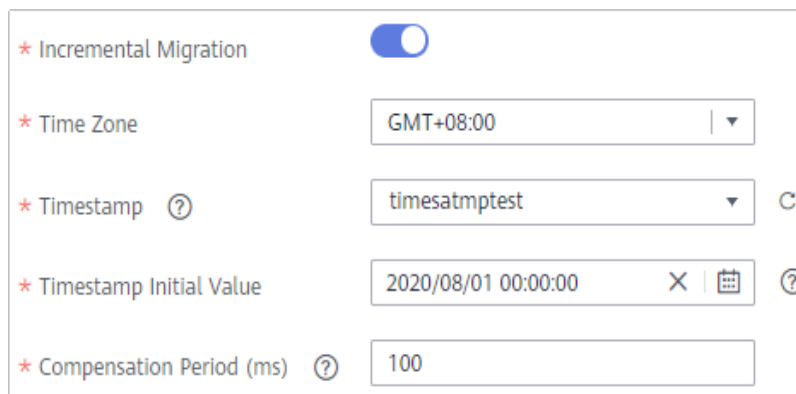
#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The preceding JSON data that contains arrays is used as an example. The following describes the configuration when the destination is MySQL:

- In the example of incremental migration configuration, the data table must contain a field of the DATE, TIME, or TIMESTAMP type as the timestamp field.

**Figure 5-49** MySQL incremental migration configuration example



The screenshot shows a configuration panel for MySQL incremental migration. It includes the following settings:

- Incremental Migration:** A toggle switch is turned on.
- Time Zone:** A dropdown menu is set to "GMT+08:00".
- Timestamp:** A dropdown menu is set to "timesatmptest".
- Timestamp Initial Value:** A text input field contains "2020/08/01 00:00:00".
- Compensation Period (ms):** A text input field contains "100".

- In the example of extended metadata configuration, the child elements **c** and **d** are obtained from the **desc** field in the data table.

**Figure 5-50** MySQL extended metadata configuration example

| Field Name | Type    | Parsing Path |
|------------|---------|--------------|
| desc       | integer | a.b.c        |
| desc       | string  | a.b.d        |

## MongoDB

[Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, MongoDB can be selected as the source data source.

**Table 5-83** MongoDB source information

| Parameter               | Description   |
|-------------------------|---|
| Table                   | Select the data set to be obtained from the MongoDB data source. (The data set is equivalent to a data table in a relational database.) Then, click <b>Select Fields in Set</b> and select only the column fields that you want to integrate and synchronize.   |
| Incremental Migration   | This parameter specifies whether only data generated in a specific period is integrated.<br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected. |
| Timestamp Field         | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.  |
| Time Zone               | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the MongoDB data source so that ROMA Connect can identify the data timestamps.  |
| Timestamp Initial Value | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |

| Parameter                    | Description   |
|------------------------------|---|
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task. This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p> |

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is MongoDB is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

## OBS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, OBS can be selected as the source data source.

**Table 5-84** OBS source information

| Parameter | Description  |
|-----------|--|
| Path      | Enter the name of the object from which data is to be obtained in the OBS data source. The value of <b>Path</b> cannot end with a slash (/). |



| Parameter        | Description   |
|------------------|---|
| File Name Prefix | Enter a file name prefix. This parameter is used together with <b>Time Format</b> to filter the data files to be integrated.  |
| Time Format      | Select the time format to be used in the file name. This parameter is used together with <b>File Name Prefix</b> to filter the data files to be integrated.   |
| File Type        | Select the format of the data file obtained from the OBS data source. The value can be <b>txt</b> , <b>csv</b> , or <b>zip</b> .  |
| Field Separator  | Enter the field separator for the data file to distinguish different fields in each row of data.  |
| Encoding Format  | Select the encoding mode of data files obtained from the OBS data source. The value can be <b>UTF-8</b> or <b>GBK</b> .   |
| Skip Title       | This parameter specifies whether to skip the title lines in the data file. The title is the first line or several lines at the beginning of a file, which helps identify and distinguish the file content.  |
| Title Lines      | Enter the number of rows in the title information in the data file so that ROMA Connect can identify the start row of the data in the file.   |
| Metadata         | This parameter specifies each data field in a data file obtained from the source to be integrated to the destination. Metadata must be filled according to the sequence of the fields in the file. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> |

The following describes the configuration when the source is OBS. The **id**, **name**, and **info** fields are obtained from the OBS data source and need to be integrated to the destination.

**Figure 5-51** OBS configuration example

Configuration

\* Path ?

File Name Prefix ?

Time Format ?

\* File Type

\* Field Separator

\* Encoding Format

\* Skip Title

\* Metadata

| Alias <span>?</span> | type   | Operation                                     |
|----------------------|--------|---|
| id                   | Double | <a href="#">Edit</a>   <a href="#">Delete</a> |
| name                 | String | <a href="#">Edit</a>   <a href="#">Delete</a> |
| info                 | String | <a href="#">Edit</a>   <a href="#">Delete</a> |
| -                    |        |   |

## Oracle

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, Oracle can be selected as the source data source.

**Table 5-85** Oracle source information

| Parameter               | Description   |
|-------------------------|---|
| Insert SQL              | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                   | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the Oracle data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>  |
| Field Sorting           | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration   | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Time Zone               | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the Oracle data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field         | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>   |
| Timestamp Initial Value | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>  |

| Parameter                    | Description  |
|------------------------------|--|
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 - 100 ms).</p>        |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata            | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"> <li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li> <li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li> <li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.  
In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is Oracle is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

## PostgreSQL

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, PostgreSQL can be selected as the source data source.

**Table 5-86** PostgreSQL source information

| Parameter             | Description   |
|-----------------------|---|
| Insert SQL            | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"><li>• If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li><li>• If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li></ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table                 | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table from which data is to be obtained in the PostgreSQL data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>  |
| Field Sorting         | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b>.</p>  |
| Incremental Migration | <p>This parameter specifies whether only data generated in a specific period is integrated.</p> <p>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.</p>  |
| Time Zone             | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select the time zone used by the PostgreSQL data source so that ROMA Connect can identify the data timestamps.</p>  |
| Timestamp Field       | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.</p>   |

| Parameter                    | Description  |
|------------------------------|--|
| Timestamp Initial Value      | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.</p>   |
| Reset Initial Migration Time | <p>This parameter can be set only when you edit an FDI task.</p> <p>This parameter specifies whether to enable the reset of the initial migration time.</p> <p>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b>.</p> <p>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.</p>  |
| Compensation Period (ms)     | <p>Mandatory when <b>Incremental Migration</b> is enabled.</p> <p>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.</p> <p>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b>, the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms).</p>        |
| Filter                       | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata            | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"> <li>• <b>Field Name:</b> name of the data field whose child elements need to be collected in the table.</li> <li>• <b>Type:</b> data type of the data element to be collected in the JSON field value.</li> <li>• <b>Parsing Path:</b> complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li> </ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

#### NOTE

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is PostgreSQL is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

## RabbitMQ

### [Back to Overview](#)

If **Integration Mode** is set to **Real-Time**, RabbitMQ can be selected as the source data source.



**Table 5-87** RabbitMQ source information

| Parameter          | Description  |
|--------------------|--|
| New Queue Creation | This parameter specifies whether to create a queue in the RabbitMQ data source. <ul style="list-style-type: none"><li>• If this parameter is set to <b>Yes</b>, a new queue is created and data is obtained from the new queue.</li><li>• If this parameter is set to <b>No</b>, data is obtained from the existing queue.</li></ul>   |
| Exchange Mode      | This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b> .<br>Select a routing mode for the exchange in RabbitMQ to forward messages to the new queue. <ul style="list-style-type: none"><li>• <b>Direct</b>: If the routing key in a message fully matches the queue, the message will be forwarded to the queue.</li><li>• <b>Topic</b>: If the routing key in a message approximately matches the queue, the message will be forwarded to the queue.</li><li>• <b>Fanout</b>: All messages will be forwarded to the queue.</li><li>• <b>Headers</b>: If the Headers attribute of a message fully matches the queue, the message will be forwarded to the queue.</li></ul> |
| Exchange Name      | This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b> .<br>Enter the exchange name of the new queue in RabbitMQ.  |
| Routing Key        | This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Direct</b> or <b>Topic</b> .<br>Enter the routing key of the new queue. RabbitMQ uses the routing key as a condition and forwards messages that meet the condition to the new queue.   |
| Message Parameters | This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Headers</b> .<br>Enter the Headers key-value pair of the new queue. RabbitMQ uses Headers as a condition and forwards messages that meet the condition to the new queue.   |
| Queue Name         | Enter the name of the message queue whose data is to be obtained. <ul style="list-style-type: none"><li>• If <b>New Queue Creation</b> is set to <b>Yes</b>, enter a new queue name.</li><li>• If <b>New Queue Creation</b> is set to <b>No</b>, enter the name of an existing queue in the RabbitMQ data source.</li></ul>  |
| Automatic Deletion | This parameter specifies whether a queue will be automatically deleted if no client is connected.  |

| Parameter   | Description  |
|-------------|--|
| Persistence | This parameter specifies whether messages in a queue are stored permanently.   |
| Metadata    | This parameter specifies each underlying key-value data element that is obtained from the source in JSON format and needs to be integrated to the destination. <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

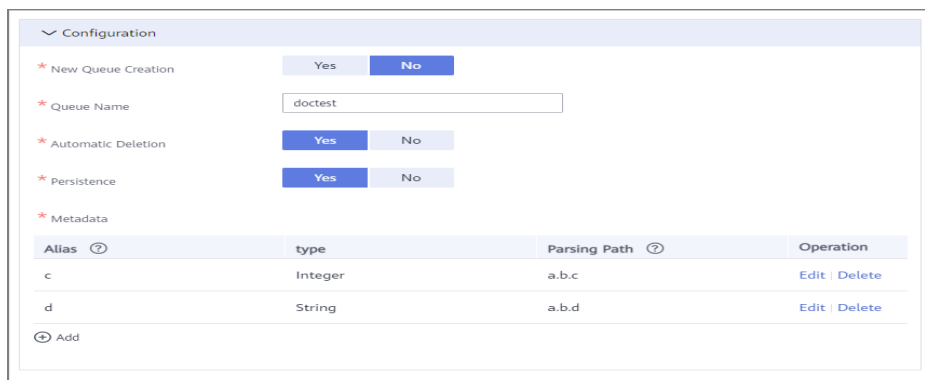
- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the source is RabbitMQ:

**Figure 5-52** RabbitMQ configuration example



## SQL Server

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, SQL Server can be selected as the source data source.

**Table 5-88** SQL Server source information

| Parameter  | Description  |
|------------|--|
| Insert SQL | <p>This parameter specifies whether SQL statements are used to obtain source data.</p> <ul style="list-style-type: none"> <li>If <b>Insert SQL</b> is enabled, ROMA Connect obtains source data based on the entered SQL statements.</li> <li>If <b>Insert SQL</b> is disabled, ROMA Connect obtains source data based on the conditions configured on the GUI.</li> </ul> <p>If <b>Insert SQL</b> is enabled, you need to enter an SQL statement for querying data. The statement must be a SELECT statement and contain the WHERE condition. The INSERT, UPDATE, DELETE, and DROP statements cannot be used. Click <b>Check SQL</b> to check the validity of the statement.</p> <p>For example, if you enter the statement <b>SELECT col01, col02 FROM table01 WHERE col02 IN('A', 'B', 'C') ORDER BY col01</b>, ROMA Connect selects the <b>col01</b> and <b>col02</b> columns from <b>table01</b>, displays data records whose values are <b>A</b>, <b>B</b>, or <b>C</b> in the <b>col02</b> column, and sorts the data records in sequence based on the values in the <b>col01</b> column.</p> |
| Table      | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Select the data table in the SQL Server data source. Then, click <b>Select Table Field</b> and select only the column fields that you want to integrate and synchronize.</p>   |

| Parameter                    | Description   |
|------------------------------|---|
| Field Sorting                | Mandatory when <b>Insert SQL</b> is disabled.<br>This parameter specifies how the data to be integrated is sorted by field. To sort the integrated data, select a reference field and select <b>Ascending</b> or <b>Descending</b> .  |
| Incremental Migration        | This parameter specifies whether only data generated in a specific period is integrated.<br>For the first scheduling, the data between the initial timestamp and the current scheduling time is collected. For subsequent scheduling, the data between the last successful collection time and the current time is collected.   |
| Time Zone                    | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select the time zone used by the SQL Server data source so that ROMA Connect can identify the data timestamps.   |
| Timestamp Field              | Mandatory when <b>Incremental Migration</b> is enabled.<br>Select a field of the DATE, TIME, or TIMESTAMP type in the data table as the timestamp of source data to determine whether the data meets the incremental integration conditions.  |
| Timestamp Initial Value      | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the time at which data is to be integrated for the first time. That is, only the data generated after this time point will be integrated.   |
| Reset Initial Migration Time | This parameter can be set only when you edit an FDI task.<br>This parameter specifies whether to enable the reset of the initial migration time.<br>Enable: The start time of each scheduling task during incremental migration is the time specified by <b>Timestamp Initial Value</b> .<br>Disable: The start time of each scheduling task during incremental migration is the end time of the previous task.   |
| Compensation Period (ms)     | Mandatory when <b>Incremental Migration</b> is enabled.<br>This parameter specifies the period of time (in milliseconds) which will be used to compensate for any delay in data generation at the source when ROMA Connect queries incremental data. The end time for obtaining data is the current system time minus the value you specify here.<br>For example, if the end time of the previous incremental migration task is 15:05, the current scheduled task is triggered at 17:00, and <b>Compensation Period (ms)</b> is set to <b>100</b> , the time range of data to be integrated in the current incremental migration task is 15:05 to (17:00 – 100 ms). |

| Parameter           | Description  |
|---------------------|--|
| Filter              | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>Add filter criteria for data to be integrated. Only the source data that meets the filter criteria will be integrated to the destination.</p> <p>For example, the condition <b>and   col02   equal   A</b> means that only the data records whose values are <b>A</b> in the <b>col02</b> column will be integrated.</p>   |
| Extended Metadata a | <p>Mandatory when <b>Insert SQL</b> is disabled.</p> <p>This parameter is mandatory if the value of a field in the database is in JSON format and the underlying key-value data elements in the JSON field value need to be collected.</p> <ul style="list-style-type: none"><li>• <b>Field Name</b>: name of the data field whose child elements need to be collected in the table.</li><li>• <b>Type</b>: data type of the data element to be collected in the JSON field value.</li><li>• <b>Parsing Path</b>: complete path of the data element in the JSON field value. For details, see <a href="#">Description on Extended Metadata Parsing Path Configuration</a>.</li></ul> |

### Description on Extended Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:  
For example, in the following JSON data, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements, that is, the data to be integrated to the destination.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
  }
}
```

```

    "c": "yy",
    "d": "yy"
  }
]
}

```

 **NOTE**

- Incremental migration does not support synchronization of physical deletion operations on source data tables to the destination. Logical deletion is recommended at the source.
- For data changes at the source, the timestamp of data rows needs to be updated synchronously. ROMA Connect compares the timestamp with the task execution time to identify the data to be incrementally migrated.

The configuration when the source is SQL Server is similar to that when the source is MySQL. For details, see [MySQL configuration example](#).

### 5.5.3 Configuring Destination Information

#### Overview

This topic describes how to configure destination information for a data integration task. Based on the destination information (including the data source and data storage), ROMA Connect writes data to the destination. The destination information configuration varies depending on data source types.

 **NOTE**

During data migration, if a primary key conflict occurs at the destination, data is automatically updated based on the primary key.

| Data Source Types Supported by Both Real-Time and Scheduled Integration Tasks   | Data Source Types Supported by Only Scheduled Tasks |
|---|---|
| <ul style="list-style-type: none"> <li>• <a href="#">API</a></li> <li>• <a href="#">ActiveMQ</a></li> <li>• <a href="#">ArtemisMQ</a></li> <li>• <a href="#">DB2</a></li> <li>• <a href="#">DWS</a></li> <li>• <a href="#">Kafka</a></li> <li>• <a href="#">MySQL</a></li> <li>• <a href="#">MongoDB</a></li> <li>• <a href="#">Oracle</a></li> <li>• <a href="#">PostgreSQL</a></li> <li>• <a href="#">RabbitMQ</a></li> <li>• <a href="#">SQL Server</a></li> </ul> | <p><a href="#">OBS</a></p>                          |

## API

API can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-89** API destination information

| Parameter          | Description  |
|--------------------|--|
| Request Parameters | <p>Construct the parameter definition of an API request. For example, the data to be integrated to the destination must be defined in <b>Body</b>. Set this parameter based on the definition of the API data source.</p> <ul style="list-style-type: none"> <li>• <b>Params</b>: parameters defined after the question mark (?) in the request URL. Only fixed values can be transferred. The method for setting <b>Params</b> is similar to that for setting <b>Body</b> in <b>form-data</b> format.</li> <li>• <b>Headers</b>: message headers of RESTful requests. Only fixed values can be transferred. The method for setting <b>Headers</b> is similar to that for setting <b>Body</b> in <b>form-data</b> format.</li> <li>• <b>Body</b>: bottom-layer parameters in the body of RESTful requests. This parameter and <b>Data Root Field</b> constitute the complete body of a request sent to the destination API. Source data is transferred to the destination API through parameters defined in <b>Body</b>. <b>Body</b> supports two modes: <b>form-data</b> and <b>raw</b>. For details, see <a href="#">Description on Body Parameter Configuration</a>.</li> </ul> |
| Data Root Field    | <p>This parameter specifies the path of upper-layer common fields in all parameters in the body sent to the destination in JSON format. <b>Data Root Field</b> and <b>Body</b> in <b>Request Parameters</b> form the request body sent to the destination API.</p> <p>For example, if the body parameter is <code>{"c":"xx","d":"xx"}</code> and <b>Data Root Field</b> is set to <b>a.b</b>, the encapsulated request data is <code>{"a":{"b":{"c":"xx","d":"xx"}}</code>.</p>  |
| Mapping Rules      | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"> <li>• <b>Mapping mode</b>: Enter the source and destination fields in a list to add their mappings.</li> <li>• <b>Script mode</b>: Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li> </ul>   |

### Description on Body Parameter Configuration

- form-data mode:  
Set **Key** to the parameter name defined by the API data source and leave **Value** empty. The key will be used as the destination field name in mapping information to map and transfer the value of the source field.

Figure 5-53 form-data mode

| Key | Value | Operation |
|-----|-------|-----------|
| c   |       | Delete    |
| d   |       | Delete    |

- Raw mode:  
The raw mode supports the JSON, Array, and nested JSON formats. Enter an example body sent to the destination API in JSON format. ROMA Connect replaces the parameter values in the example based on the mapping configuration, and finally transfers the source data to the destination. The following is an example body in raw mode:

- JSON format:

```
{
  "id": 1,
  "name": "name1"
}
```

Enter the body in JSON format, leave **Data Root Field** empty, and set the field names in **Mapping Information**.

| Destination Field | Source Field |
|-------------------|--------------|
| id                | id           |
| name              | name         |

- Array format:

```
{
  "record": [
```



```
{  
  "id": 1,  
  "name": ""  
}
```

Set **Data Root Field** to the JSONArray object name, for example, **record**. Enter field names in mapping information.

The screenshot displays the 'Request Parameters' section with the 'Body' tab selected. The 'raw' radio button is chosen, and a red box highlights the JSON body content: 

```
{  
  "record": [  
    {  
      "id": 1,  
      "name": ""  
    }  
  ]  
}
```

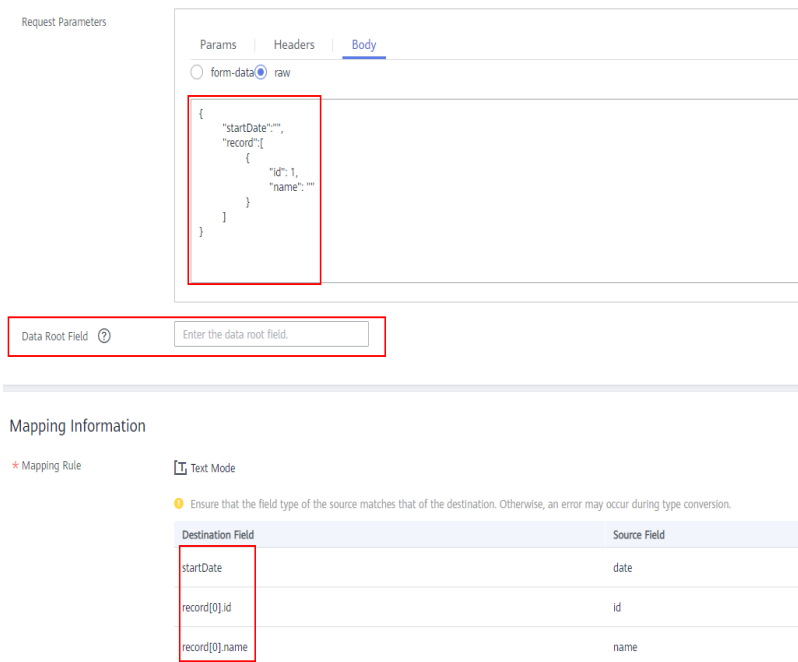
. Below this, the 'Data Root Field' is set to 'record'. The 'Mapping Information' section is also visible, showing a table with two columns: 'Destination Field' and 'Source Field'. The 'id' and 'name' fields are mapped from source to destination, with a red box highlighting the 'id' row.

| Destination Field | Source Field |
|-------------------|--------------|
| id                | id           |
| name              | name         |

– **Nested JSON format:**

```
{  
  "startDate": "",  
  "record": [  
    {  
      "id": 1,  
      "name": ""  
    }  
  ]  
}
```

Leave **Data Root Field** blank. In mapping information, set the json fields to the field names and set the jsonArray fields to specific paths, for example, **record[0].id**.



## ActiveMQ

[Back to Overview](#)

ActiveMQ can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-90** ActiveMQ destination information

| Parameter        | Description   |
|------------------|---|
| Destination Type | Select the message transfer model of the ActiveMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name | Enter the name of a topic or queue to which to send data. Ensure that the topic or queue already exists.  |
| Metadata         | <p>Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"> <li>● <b>Alias</b>: user-defined metadata name.</li> <li>● <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li> <li>● <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li> </ul> |

| Parameter     | Description   |
|---------------|---|
| Mapping Rules | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the destination is ActiveMQ:

**Figure 5-54** ActiveMQ configuration example

The screenshot shows a configuration form for ActiveMQ. It includes a 'Destination Type' dropdown set to 'Topic', a 'Destination Name' text input containing 'doctest', and a 'Metadata' table. The table has columns for 'Alias', 'Type', 'Parsing Path', and 'Operation'. Two rows are visible: one with Alias 'c', Type 'Integer', and Parsing Path 'a.b.c'; another with Alias 'd', Type 'String', and Parsing Path 'a.b.d'. Each row has 'Edit' and 'Delete' links. An 'Add' button is at the bottom.

| * Destination Type   | Topic   |                |             |
|----------------------|---------|----------------|-------------|
| * Destination Name ⓘ | doctest |                |             |
| * Metadata           |         |                |             |
| Alias ⓘ              | Type    | Parsing Path ⓘ | Operation   |
| c                    | Integer | a.b.c          | Edit Delete |
| d                    | String  | a.b.d          | Edit Delete |
| ⊕ Add                |         |                |             |

## ArtemisMQ

### [Back to Overview](#)

ArtemisMQ can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-91** ArtemisMQ destination information

| Parameter         | Description  |
|-------------------|--|
| Destination Type  | Select the message transfer model of the ArtemisMQ data source. The value can be <b>Topic</b> or <b>Queue</b> .  |
| Destination Name  | Enter the name of a topic or queue to which to send data. Ensure that the topic or queue already exists.   |
| Extended Metadata | Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>● <b>Alias</b>: user-defined metadata name.</li><li>● <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>● <b>Parsing Path</b>: full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |
| Mapping Rules     | Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a> . <ul style="list-style-type: none"><li>● <b>Mapping mode</b>: Enter the source and destination fields in a list to add their mappings.</li><li>● <b>Script mode</b>: Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul>  |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [
      {
        "c": "xx",
        "d": "xx"
      },
      {
        "c": "yy",
        "d": "yy"
      }
    ]
  }
}
```

The configuration when the destination is ArtemisMQ is similar to that when the destination is ActiveMQ. For details, see [ActiveMQ configuration example](#).

## DB2

### [Back to Overview](#)

DB2 can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-92** DB2 destination information

| Parameter | Description  |
|-----------|--|
| Table     | Select the data table to which data will be written. Then, click <b>Select Table Field</b> and select only the column fields that you want to write. |

| Parameter           | Description   |
|---------------------|---|
| Batch Number Field  | <p>Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.</p> <p>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback.</p>  |
| Batch Number Format | <p>The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.</p>  |
| Mapping Rules       | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

## DWS

### [Back to Overview](#)

DWS can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-93** DWS destination information

| Parameter          | Description  |
|--------------------|--|
| Table              | <p>Select the data table to which data will be written. Then, click <b>Select Table Field</b> and select only the table fields that you want to write.</p>   |
| Batch Number Field | <p>Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.</p> <p>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback.</p> |

| Parameter           | Description   |
|---------------------|---|
| Batch Number Format | The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Mapping Rules       | Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a> . <ul style="list-style-type: none"><li>● <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>● <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

## Kafka

### [Back to Overview](#)

Kafka can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-94** Kafka destination information

| Parameter  | Description   |
|------------|---|
| Topic Name | Select the name of the topic to which data is to be written.  |
| Key        | Enter the key value of a message so that the message will be stored in a specified partition. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner.   |
| Metadata   | Define the data fields to be written to the destination Kafka. The number of fields to be integrated at the source determines the number of metadata records defined on the destination. <ul style="list-style-type: none"><li>● <b>Alias:</b> user-defined metadata name.</li><li>● <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul> |

| Parameter     | Description  |
|---------------|--|
| Mapping Rules | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"> <li>● <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li> <li>● <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li> </ul> |

The following figure shows a configuration example when the destination is Kafka. **id**, **name**, and **info** are the data fields to be written to the Kafka data source.

**Figure 5-55** Kafka configuration example

The screenshot shows a configuration interface for Kafka. It includes the following elements:

- \* Topic Name:** A text input field containing "topic-doc".
- Key:** A text input field with the placeholder "Enter a key".
- \* Metadata:** A section with a "Text Mode" icon and a table of mapping rules.

| Alias | Type    | Operation     |
|-------|---------|---------------|
| id    | Integer | Edit   Delete |
| name  | String  | Edit   Delete |
| info  | String  | Edit   Delete |

At the bottom of the metadata section, there is an "Add" button.

The structure of the message written to Kafka is {"id": "xx", "name": "yy", "info": "zz"}, where xx, yy, and zz are the data values transferred from the source.

## MySQL

### [Back to Overview](#)

MySQL can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-95** MySQL destination information

| Parameter | Description   |
|-----------|---|
| Table     | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate. |



| Parameter           | Description   |
|---------------------|---|
| Batch Number Field  | <p>Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.</p> <p>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback.</p>  |
| Batch Number Format | <p>The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.</p>  |
| Mapping Rules       | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>● <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>● <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

## MongoDB

### [Back to Overview](#)

MongoDB can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-96** MongoDB destination information

| Parameter  | Description   |
|------------|---|
| Table      | <p>Select the data set to be written to the MongoDB data source. (The data set is equivalent to a data table in a relational database.) Then, click <b>Select Fields in Set</b> and select only the column fields that you want to write.</p> |
| Upsert     | <p>This parameter indicates whether to update or insert data to the destination, that is, whether to directly updating existing data fields in the data set on the destination.</p>   |
| Upsert key | <p>This parameter is mandatory only if <b>Upsert</b> is enabled.<br/>Select the data field to be upserted.</p>  |

| Parameter     | Description   |
|---------------|---|
| Mapping Rules | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

## OBS

### [Back to Overview](#)

If **Integration Mode** is set to **Scheduled**, OBS can be selected as the destination data source.

**Table 5-97** OBS destination information

| Parameter           | Description  |
|---------------------|--|
| Object Type         | Select the type of the data file to be written to the OBS data source. Currently, <b>Text file</b> and <b>Binary file</b> are supported.   |
| Encoding Format     | This parameter is mandatory only if <b>Object Type</b> is set to <b>Text file</b> .<br>Select the encoding mode of the data file to be written to the OBS data source. The value can be <b>UTF-8</b> or <b>GBK</b> . |
| Path                | Enter the name of the object to be written to the OBS data source. The value of <b>Path</b> cannot end with a slash (/).   |
| File Name Prefix    | Enter the prefix of the file name. This parameter is used together with <b>Time Format</b> to define the name of the file to be written to the OBS data source.  |
| Time Format         | Select the time format to be used in the file name. This parameter is used together with <b>File Name Prefix</b> to define the data file to be written to the OBS data source.                                       |
| File Type           | Select the format of the data file to be written to the OBS data source. A text file can be in TXT or CSV format, and a binary file can be in XLS or XLSX format.  |
| Advanced Attributes | This parameter is mandatory only if <b>File Type</b> is set to <b>csv</b> .<br>Select whether to configure the advanced properties of the file.  |

| Parameter           | Description  |
|---------------------|--|
| Newline             | <p>This parameter is mandatory only if <b>Advanced Attributes</b> is set to <b>Enable</b>.</p> <p>Enter a newline character in the file content to distinguish different data lines in the file.</p>   |
| Enclosure Character | <p>This parameter is mandatory only if <b>Advanced Attributes</b> is set to <b>Enable</b>.</p> <p>If you select <b>Use</b>, each data field in the data file is enclosed by double quotation marks ("). If a data field contains the same symbol as a separator or newline character, the field will not be split into two fields. For example, if source data contains a data field <b>aa bb</b> and the vertical bar ( ) is set as the separator when the source data is integrated to the destination data file, the field is <b>aa bb</b> in the destination data file and will not be split into <b>aa</b> and <b>bb</b>.</p> |
| Field Separator     | <p>This parameter is mandatory only if <b>File Type</b> is set to <b>txt</b> or <b>Advanced Attributes</b> is set to <b>Enable</b>.</p> <p>Enter the field separator for the file contents to distinguish different fields in each row of data.</p>  |
| Add File Header     | <p>Determine whether to add a file header to the data file to be written. The file header is the first line or several lines at the beginning of a file, which helps identify and distinguish the file content.</p>  |
| File Header         | <p>This parameter is mandatory only if <b>Add File Header</b> is set to <b>Yes</b>.</p> <p>Enter the file header information. Use commas (,) to separate multiple file headers.</p>  |
| Metadata            | <p>Define the data fields to be written to the destination file. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias</b>: user-defined metadata name.</li><li>• <b>Type</b>: data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li></ul>  |
| Mapping Rules       | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>• <b>Mapping mode</b>: Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode</b>: Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul>  |

The following figure shows a configuration example when the destination is OBS. **id**, **name**, and **info** are the data fields to be written to the OBS data source.

**Figure 5-56** OBS configuration example

The screenshot shows a configuration form for OBS. It includes the following fields and values:

- Object Type:** Text file
- Encoding Format:** UTF-8
- Path:** data/test
- File Name Prefix:** roma
- Time Format:** yyyy-MM-dd
- File Type:** txt
- Field Separator:** ,
- Add File Header:** No
- Metadata:** Text Mode

| Alias ? | Type   |
|---------|--------|
| id      | Double |
| name    | String |
| info    | String |

## Oracle

### [Back to Overview](#)

Oracle can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-98** Oracle destination information

| Parameter | Description   |
|-----------|---|
| Table     | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate. |

| Parameter           | Description   |
|---------------------|---|
| Batch Number Field  | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback.   |
| Batch Number Format | The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.   |
| Mapping Rules       | Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a> . <ul style="list-style-type: none"><li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

 **NOTE**

For the Oracle destination data source, if the source primary key field is empty, the record is discarded by default and no scheduling log error code is generated.

## PostgreSQL

### [Back to Overview](#)

PostgreSQL can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-99** PostgreSQL destination information

| Parameter         | Description  |
|-------------------|--|
| Auto Create Table | Specify whether to automatically create a data table.  |
| Destination Table | This parameter is displayed only when <b>Auto Create Table</b> is enabled.<br><br>Enter the name of the table to be automatically created. |

| Parameter           | Description   |
|---------------------|---|
| Table               | <p>This parameter is displayed only when <b>Auto Create Table</b> is disabled.</p> <p>Select the data table to which data is to be written and click <b>Select Table Field</b> to select the data column fields to be integrated.</p>   |
| Batch Number Field  | <p>Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.</p> <p>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback.</p>  |
| Batch Number Format | <p>The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.</p>  |
| Mapping Rules       | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>● <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>● <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul> |

## RabbitMQ

### [Back to Overview](#)

RabbitMQ can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-100** RabbitMQ destination information

| Parameter          | Description   |
|--------------------|---|
| New Queue Creation | <p>Determine whether to create a queue in the RabbitMQ data source.</p> <ul style="list-style-type: none"><li>● If you select <b>Yes</b>, a new queue is created and the data to be integrated is sent to the queue.</li><li>● If you select <b>No</b>, the data to be integrated is sent to an existing queue.</li></ul> |

| Parameter          | Description   |
|--------------------|---|
| Exchange Mode      | <p>Select a routing mode for the exchange in the RabbitMQ data source to forward messages to the queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, select the routing mode for the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, select the routing mode that is the same as that of the existing destination queue.</p> <ul style="list-style-type: none"><li>• <b>Direct</b>: If the routing key in a message fully matches the queue, the message will be forwarded to the queue.</li><li>• <b>Topic</b>: If the routing key in a message approximately matches the queue, the message will be forwarded to the queue.</li><li>• <b>Fanout</b>: All messages will be forwarded to the queue.</li><li>• <b>Headers</b>: If the Headers attribute of a message fully matches the queue, the message will be forwarded to the queue.</li></ul> |
| Exchange Name      | <p>Enter the exchange name of the RabbitMQ data source. If <b>New Queue Creation</b> is set to <b>Yes</b>, the exchange name of the new queue is used. If <b>New Queue Creation</b> is set to <b>No</b>, configure the exchange name that is the same as that of the existing destination queue.</p>  |
| Routing Key        | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Direct</b> or <b>Topic</b>.</p> <p>RabbitMQ uses the routing key as the judgment condition. Messages that meet the condition will be forwarded to the queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, enter the routing key of the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, enter the routing key that is the same as that of the existing destination queue.</p>  |
| Message Parameters | <p>This parameter is mandatory only if <b>Exchange Mode</b> is set to <b>Headers</b>.</p> <p>RabbitMQ uses Headers as a judgment condition. Messages that meet the condition will be forwarded to a new queue. If <b>New Queue Creation</b> is set to <b>Yes</b>, enter the headers of the new queue. If <b>New Queue Creation</b> is set to <b>No</b>, enter the headers that are the same as those of the existing destination queue.</p>   |
| Queue Name         | <p>This parameter is mandatory only if <b>New Queue Creation</b> is set to <b>Yes</b>.</p> <p>Enter the name of a new queue.</p>  |
| Automatic Deletion | <p>This parameter specifies whether a queue will be automatically deleted if no client is connected.</p>  |
| Persistence        | <p>This parameter specifies whether messages in a queue are stored permanently.</p>   |

| Parameter     | Description   |
|---------------|---|
| Metadata      | <p>Define each underlying key-value data element to be written to the destination in JSON format. The number of fields to be integrated at the source determines the number of metadata records defined on the destination.</p> <ul style="list-style-type: none"><li>• <b>Alias:</b> user-defined metadata name.</li><li>• <b>Type:</b> data type of metadata. The value must be the same as the data type of the corresponding parameter in the source data.</li><li>• <b>Parsing Path:</b> full path of metadata. For details, see <a href="#">Description on Metadata Parsing Path Configuration</a>.</li></ul> |
| Mapping Rules | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"><li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li><li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li></ul>   |

### Description on Metadata Parsing Path Configuration

- Data in JSON format does not contain arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b.c**, and **a.b.d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b.c**, and **Parsing Path** of element d must be set to **a.b.d**.

```
{
  "a": {
    "b": {
      "c": "xx",
      "d": "xx"
    }
  }
}
```

- Data in JSON format contains arrays:

For example, in the following JSON data written to the destination, the complete paths for elements a to d are **a**, **a.b**, **a.b[i].c**, and **a.b[i].d**, respectively. Elements c and d are underlying data elements.

In this scenario, **Parsing Path** of element c must be set to **a.b[i].c**, and **Parsing Path** of element d must be set to **a.b[i].d**.

```
{
  "a": {
    "b": [{
      "c": "xx",
      "d": "xx"
    }],
  }
}
```



```

    "c": "yy",
    "d": "yy"
  }
]
}

```

The preceding JSON data that does not contain arrays is used as an example. The following describes the configuration when the destination is RabbitMQ:

**Figure 5-57** RabbitMQ configuration example

The screenshot shows a configuration form for RabbitMQ. It includes several sections:

- \* New Queue Creation:** Radio buttons for 'Yes' and 'No'.
- \* Exchange Mode:** A dropdown menu set to 'Direct'.
- \* Exchange Name:** A text input field containing 'doctest'.
- \* Routing Key:** A text input field containing 'roma'.
- \* Automatic Deletion:** Radio buttons for 'Yes' and 'No'.
- \* Persistency:** Radio buttons for 'Yes' and 'No'.
- \* Metadata:** A table with columns 'Alias', 'Type', and 'Parsing Path'.
 

| Alias | Type    | Parsing Path |
|-------|---------|--------------|
| c     | Integer | a.b.c        |
| d     | String  | a.b.d        |

## SQL Server

[Back to Overview](#)

SQL Server can be selected as the destination data source when **Integration Mode** is set to **Scheduled** or **Real-Time**.

**Table 5-101** SQL Server information at the destination

| Parameter           | Description   |
|---------------------|---|
| Table               | Select an existing table and click <b>Select Table Field</b> to select only the column fields that you want to integrate.   |
| Batch Number Field  | Select a field whose type is <b>String</b> and length is greater than 14 characters in the destination table. In addition, the batch number field cannot be the same as the destination field in mapping information.<br><br>The value of this field is a random number, which is used to identify the data in the same batch. The data inserted in the same batch uses the same batch number, indicating that the data is inserted in the same batch and can be used for location or rollback. |
| Batch Number Format | The batch number can be in yyyyMMddHHmmss or UUID format. You are advised to use the UUID format because batch numbers in yyyyMMddHHmmss format are not unique.   |

| Parameter     | Description  |
|---------------|--|
| Mapping Rules | <p>Configure the mapping rules between source data fields and destination data fields to convert the source data obtained into the destination data. For details about the mapping, see <a href="#">Configuring Mapping Rules</a>.</p> <ul style="list-style-type: none"> <li>• <b>Mapping mode:</b> Enter the source and destination fields in a list to add their mappings.</li> <li>• <b>Script mode:</b> Read source data and write the data to the destination using JavaScript scripts. For details, see the <a href="#">script mapping examples</a>.</li> </ul> |

## 5.5.4 Configuring Mapping Rules

### Overview

This topic describes how to configure mapping rules for a data integration task. Configure the mapping rules between source and destination data fields so that the source data obtained can be converted to destination data, in either **Mapping** mode or **Script** mode. The **Mapping** mode can be automatic or manual.

 **NOTE**

Do not use the keywords of corresponding databases as source/destination field names. Otherwise, data integration tasks may fail.

### Mapping Mode

- **Automatic mapping**

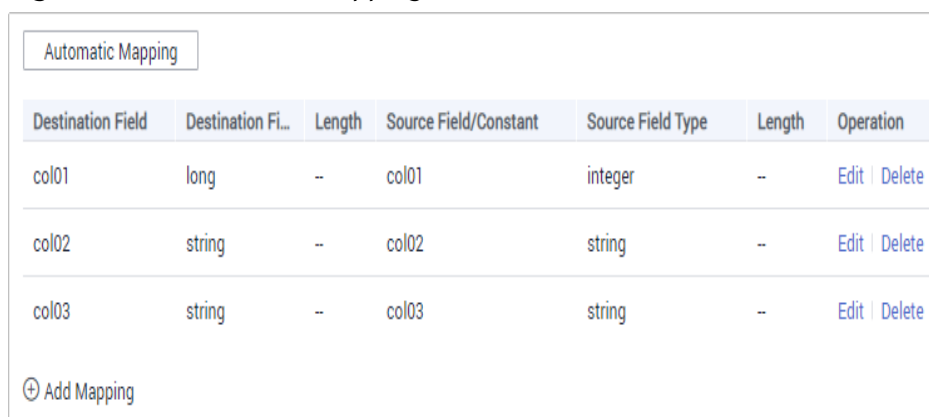
If metadata is defined at both the source and destination, you can use the automatic mapping mode to configure mapping information.

Click **Automatic Mapping**. A mapping rule between the source and destination data fields is automatically created.


 **NOTE**

If API is used as the data source type at the source or destination, data fields do not support automatic mapping. You must manually configure a mapping for data fields.

**Figure 5-58** Automatic mapping



| Destination Field | Destination Fi... | Length | Source Field/Constant | Source Field Type | Length | Operation                                     |
|-------------------|-------------------|--------|-----------------------|-------------------|--------|---|
| col01             | long              | --     | col01                 | integer           | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |
| col02             | string            | --     | col02                 | string            | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |
| col03             | string            | --     | col03                 | string            | --     | <a href="#">Edit</a>   <a href="#">Delete</a> |

 Add Mapping

- **Manual mapping**

If the data table fields at the source and at the destination are inconsistent, click **Add** to add a new mapping and configure its rule manually. This method applies to integration for all data types. You can configure a mapping rule by entering a key-value pair or entering a value in the text box.

## Script Mode

Configure the mapping between source and destination data using JavaScript scripts when complex objects are involved. **sourceObj** indicates the field object set of the source table and **targetObj** indicates that of the destination table.

### Script mapping examples:

- **Templates**

```
function excute(sourceObj){
//Enter the script content (case sensitive).
//Define the destination data objects.
targetObj = {};
//Add, subtract, multiply, and divide operations are supported.
targetObj.a= (sourceObj.id * 3 + 1) % 5;
//The JavaScript Math function is supported.
targetObj.b= Math.sqrt(100);
//Type conversion is supported.
targetObj.c = Number("3.14");
//The system time function can be called.
targetObj.date = new Date().toLocaleString();
//Regular expressions are supported.
targetObj.fdi = sourceObj.name.replace(/world/i,"fdi");
//JSON conversion is supported.
var json = JSON.parse(sourceObj.infoJson);
targetObj.address = json.address;
targetObj.age = json.age;
targetObj.sex = json.other.sex;
targetObj.hobby = json.other.hobby;
//Conditional statements are supported.
if(targetObj.hobby == "rap"){
targetObj.ikun = true;
}else{
targetObj.ikun = false;
}
return targetObj;
}
```

- **Mapping between fields only**

```
function excute(sourceObject) {//sourceObject indicates the data object transmitted from the source.
//Write the script content.
var targetObject = {};
targetObject.mqs_id = sourceObject.id;//Map the id field in the source table to the mqs_id field in
the destination table.
targetObject.mqs_name = sourceObject.name;
targetObject.mqs_date = sourceObject.date;
targetObject.mqs_date = sourceObject['customized-outdutydate'];//Attributes containing hyphens (-)
must be quoted using square brackets.
return targetObject; //targetObject indicates the data object returned to the destination.
}
```

- **API as the destination data source type**

When the destination data source type is API, three objects are required for transmission: **params**, **headers**, and **body**.

```
function excute(sourceObj) {
print("execute js");
print(sourceObj);
var targetObj = {};
targetObj.params = {};
```

```
targetObj.params.id = sourceObj.uid
targetObj.headers = {};
targetObj.headers['X-HW-ID'] = "ApplicationID";
targetObj.headers['X-HW-APPKEY'] = "AppKey";
targetObj.body = {};
return obj;
}
```

If the data to be converted involves time data, modify the JS script by referring to the following content in bold.

#### NOTE

During orchestration, if the destination data source type is API and the JS script is used for data conversion, source data of the Date type will be automatically converted to the yyyy-MM-dd HH:mm:ss string type. For example, if the source is **19:55:20**, the time will be converted to **1970-01-01 19:55:20** by the JS script. **1970-01-01** is the default value added by the system.

Therefore, to obtain time data in desired format, perform the conversion in the JS script by referring to the example below.

```
function excute(sourceObj) {
/**
 * Custom_time_format_function
 * @param {format} Time_display_format
 */
Date.prototype.format = function (format) {
    var date = {
        "M+": this.getMonth() + 1,
        "d+": this.getDate(),
        "h+": this.getHours(),
        "m+": this.getMinutes(),
        "s+": this.getSeconds(),
        "q+": Math.floor((this.getMonth() + 3) / 3),
        "S+": this.getMilliseconds()
    };
    if (/+(y+)/i.test(format)) {
        format = format.replace(RegExp.$1, (this.getFullYear() + "").substr(4 - RegExp.$1.length));
    }
    for (var k in date) {
        if (new RegExp("(" + k + ")").test(format)) {
            format = format.replace(RegExp.$1, RegExp.$1.length == 1 ? date[k] : ("00" + date[k]).substr(("" + date[k]).length));
        }
    }
    return format;
};

print("execute js");
print(sourceObj);
var targetObj = {};
targetObj.params = {};
targetObj.params.id = sourceObj.uid
//When the time data is converted, the custom function above is called to extract the time data (hour, minute, and second) for the destination field.
//HH:mm:ss in the following code indicates the hour, minute, and second, respectively. If the format of hhmss is used, the value returned to the destination field does not contain colons (:).
targetObj.params.time = (new Date(sourceObj.time)).format("hh:mm:ss");
targetObj.headers = {};
targetObj.headers['X-HW-ID'] = "ApplicationID";
targetObj.headers['X-HW-APPKEY'] = "AppKey";
targetObj.body = {};
return obj;
}
```

- Database fields involving multi-level JSON parsing

If multi-level JSON objects are nested in the source data object fields, use [] to reference each parsed field.

In the following example script, the **root** field contains multi-level objects such as **id**, **name**, and **double**. In this case, each parsed field needs to be referenced separately.

```
function excute(sourceObject) {
  //Write the script content.
  var targetObject = {};
  targetObject.mqs_id = sourceObject["root.id"];
  targetObject.mqs_name = sourceObject["root.name"];
  targetObject.mqs_double = sourceObject["root.double"];
  targetObject.mqs_date = sourceObject["root.date"];
  targetObject.mqs_boolean = sourceObject["root.boolean"];
  targetObject.mqs_timestamp = sourceObject["root.timestamp"];
  targetObject.mqs_time = sourceObject["root.time"];
  targetObject.mqs_long = sourceObject["root.long"];
  return targetObject;
}
```

## 5.6 Starting or Stopping a Data Integration Task

### Overview

After you create a data integration task, **Task Status** is displayed as **Stopped** by default. You need to manually start the task by clicking **Start**.

- After you start a scheduled task, ROMA Connect integrates data on a scheduled basis. During the first execution, all source data that meets the conditions is integrated to the destination. Then, full data that meets the conditions or only incremental data will be integrated based on the task configuration.
- After you start a real-time task, ROMA Connect continuously detects data changes at the source. During the first execution, all source data that meets the conditions is integrated to the destination. Subsequently, only new data will be integrated to the destination each time.

#### NOTE

- If two data integration tasks use MRS data sources of different versions (including MRS Hive, MRS HDFS, and MRS HBase) and Kerberos authentication is enabled for the MRS data sources, the two data integration tasks cannot be executed at the same time. Otherwise, the integration tasks fail.
- In the task list, **Executed** indicates the time when the schedule of a created task starts to take effect.

If you need to perform other operations, such as modification, on a started task, you must stop the task first.

### Starting a Task

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**.
3. On the **Task Management** page, filter the task to be started based on the integration mode, task status, or task name.

You can start the task by using either of the following methods:

- **Clicking Start:** The task runs according to the configured integration mode.

Select the task to be started and click **Start** above the task list. After the task is started, **Task Status** changes to **Started**.

- **Selecting Manual Scheduling:** Before manually scheduling a task, ensure that **Task Status** is **Stopped**.

Choose **More > Manual Scheduling** on the right of the data integration task to manually schedule the task.

- After a scheduled task is manually scheduled, the task is executed only once, and then the task status changes to **Stopped**.
- After a real-time task is manually scheduled, the task is started and the task status changes to **Started**. The effect of manually scheduling a real-time task is the same as that of starting a scheduled task directly.

## Stopping a Task

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**.
3. On the **Task Management** page, filter the task to be stopped based on the integration mode, task status, or task name.
4. Select the task and click **Stop** to stop the task.

## 5.7 Managing a Data Integration Task

### 5.7.1 Viewing Data Integration Tasks

#### Overview

After you create a data integration task, you can view the task information, including the task configuration information, run logs, operation logs, and scheduling logs.

You can also modify the task configuration. The procedure for modifying a task is similar to that for creating a task. For details, see [Creating a Data Integration Task](#).

---

#### NOTICE

Exercise caution when modifying task configuration information to avoid dirty data or incomplete data integration at the destination. Especially, modifying source or destination information may cause changes to the data integration solution.

---

## Viewing Task Information

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **Fast Data Integration > Task Management**.
3. On the **Task Management** page, click a task name. On the task details page, view related configuration and log information.

You can filter tasks by **Integration Mode**, **Task Status**, **Task Name**, or other criteria. Flow tasks do not have the task information page.

**Table 5-102** Task information



| Task Information  | Description   |
|-------------------|---|
| Basic information | Basic task information, including the task name, task ID, integration mode, and task status.  |
| Details           | Task configuration information, including the source, destination, and mapping information. (No mapping information is displayed for composite tasks.)  |
| Scheduling        | Schedule information of a task. Only scheduled tasks have schedule information. For real-time tasks, no schedule information is involved.   |
| Logs              | Scheduling logs and operation logs of tasks. The lasted 10,000 execution records can be retained. <ul style="list-style-type: none"><li>• A scheduling log records information about the task execution, including the schedule execution time, data reading time, data volume, duration of each phase, data source reading duration, system processing duration, and statistics period.</li><li>• An operation log records the management operations of tasks, including creating, updating, deleting, starting, and stopping tasks.</li></ul> |

## Viewing Scheduling Details

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation tree on the left, choose **Fast Data Integration > Schedule Monitoring**.
3. On the **Schedule Monitoring** page, filter information such as the running status, scheduling mode, scheduling period, and latest execution time based on the task status and task name.

For a composite task, you can also view the synchronization progress, time for reading full data in the table, and other information.

4. Click **View Log** on a task to view the detailed scheduling logs of the task.
  - You can specify **Time Range** to view scheduling logs in a specified period.

- You can click  (detail mode) or  (list mode) on the right to switch the log display mode.

## 5.7.2 Importing or Exporting Data Integration Tasks

### Overview

You can import or export data integration tasks.

### Exporting Data Integration Tasks

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation tree on the left, choose **Fast Data Integration > Task Management**.
3. On the **Task Management** page, select the required tasks and click **Export Task**. The system automatically exports the selected tasks to the local PC.

### Importing Data Integration Tasks

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation tree on the left, choose **Fast Data Integration > Task Management**.
3. On the **Task Management** page, click **Import Task** and select the file to import.
4. In the dialog box displayed, specify the import type. Options are **Global** and **Overlay**.

## 5.7.3 Appendix: Quartz Cron Expression Configuration

### Parameter Description

Quartz cron expressions can be used to schedule data integration tasks.

A Quartz cron expression contains six fields separated by space, in the following format.

**Second Minute Hour Day Month Week**

[Table 5-103](#) describes the parameters.



**Table 5-103** Quartz cron parameters

| Name   | Mandatory             | Optional Value  | Optional Special Character |
|--------|-----------------------|---|----------------------------|
| Second | No<br>To be supported | Fixed at <b>0</b>   | -                          |
| Minute | Yes                   | 0-59  | , - * /                    |
| Hour   | Yes                   | 0-23  | , - * /                    |
| Day    | Yes                   | Day of a month, ranging from 1 to 31. Note that there are no 30 and 31 in February, and there is no 31 in April, June, September, and November.   | , - * ? / L W              |
| Month  | Yes                   | 1-12 or a character string. For the mapping between numbers and character strings, see <a href="#">Table 5-104</a> .<br>Example: Enter <b>1</b> or <b>JAN</b> if you want to specify January.   | , - * /                    |
| Week   | Yes                   | 1-7 or a character string. For details about the mapping between numbers and character strings, see <a href="#">Table 5-105</a> . In Quartz cron expressions, Saturday is the last day of a week.<br>Example: Enter <b>2</b> or <b>MON</b> if you want to specify Monday. | , - * ? / L #              |

**Table 5-104** Mapping between numbers and character strings for the month

| Number | String |
|--------|--------|
| 1      | JAN    |
| 2      | FEB    |
| 3      | MAR    |
| 4      | APR    |
| 5      | MAY    |
| 6      | JUN    |
| 7      | JUL    |

| Number | String |
|--------|--------|
| 8      | AUG    |
| 9      | SEP    |
| 10     | OCT    |
| 11     | NOV    |
| 12     | DEC    |

**Table 5-105** Mapping between numbers and character strings for the week

| Number | String | Description |
|--------|--------|-------------|
| 1      | SUN    | Sunday      |
| 2      | MON    | Monday      |
| 3      | TUE    | Tuesday     |
| 4      | WED    | Wednesday   |
| 5      | THU    | Thursday    |
| 6      | FRI    | Friday      |
| 7      | SAT    | Saturday    |

## Precautions

Each parameter value is a number, but the following special characters can also be used:

- \*: It can match any value of a field. If \* is used in the minutes field, an event is triggered every minute.
- ?: It can only be used in two fields: day (day in each month) and week. It can also match any value of the field, but actually it does not. This is because day (day in each month) and week are mutually affected. For example, if you want to trigger scheduling on the 20th day of each month, you can use the following expression: 13 13 15 20 \* ?, where the last digit can only be ?, but not \*. If you use \*, an event will be triggered regardless of the day of week.
- -: It specifies a range. For example, **5-20** in the minutes field means that an event is triggered every minute from minutes 5-20.
- /: An event is triggered from the specified time and at a fixed interval. For example, if you set the minutes field to **5/20**, it means that the scheduled task will be triggered at the 5th, 25th, and 45th minutes of each hour.
- ,: It specifies a list value. For example, if you set the minutes field to **5,20**, it means that the scheduled task will be triggered every minute between minutes 5 and 20 of each hour.

- L: It indicates "last". It can appear only in the day (day in each month) and week fields. For example, **5L** in the week field means that a task is triggered on the last Thursday of the month.
- W: It indicates the weekday (Monday to Friday) nearest the given day. It can appear only in the day (day of month) field. For example, **5W** in the day (day of month) field means that a task is triggered on the nearest weekday to the 5th day of the month. If the 5th day is Saturday, the task is triggered on the 4th day (Friday). If the 5th day is Sunday, the task is triggered on the 6th day (Monday). If the 5th day is one day from Monday to Friday, an event is triggered on the 5th day. The trigger will not jump over the boundary of a month's days.
- LW: The two characters indicate the last weekday of a month, that is, the last Friday.
- #: It indicates the *n*th day of a week every month. It can appear only in the day (day of month) field. For example, **4#2** indicates the second Wednesday of a month.

## Configuration Example

The following examples show you how to use Quartz cron expressions.

### Simple examples:

```
0 0/1 * * * ? //Executed every one minute
0 0 23 * * ? //Executed at 23:00 every day
0 0 1 1 * ? //Executed at 01:00 on the first day of each month
0 0 23 L * ? //Executed at 23:00 on the last day of each month
0 33,55 * * * ? //Executed at the 33th and 55th minute
0 0 18 * * LW //Executed at 06:00 on the last weekday of each month
```

### Common examples:

```
0 0 12 * * ? //Executed at 12:00 every day
0 * 14 * * ? //Executed every minute from 02:00 to 02:59 every day
0 0/30 9-19 * * ? //Executed every half an hour from 9:00 a.m. to 19:59 p.m.
0 0 12 ? * WED //Executed at 12:00 every Wednesday
0 0/30 9-19 * * MON-FRI //Executed every half an hour from 09:00 to 19:59 every day from Monday to Friday
0 15 10 ? * MON-FRI //Executed at 10:15 every day from Monday to Friday
0 0 2 1 * ? //Executed at 02:00 on the first day of each month
0 15 10 ? * 6#3 //Executed at 10:15 on the third Friday of each month
0 15 10 ? * 6L //Executed at 10:15 on the last Friday of each month
```

## 5.8 Connectors

### 5.8.1 Creating a Connector

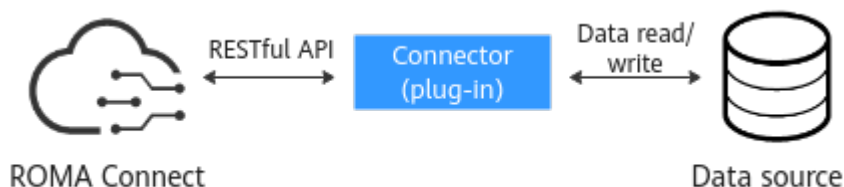
#### Overview

ROMA Connect supports connections of common data source types, such as relational databases, message queues, APIs, NoSQL databases, and OBS. These data sources can be directly used in ROMA Connect. You can integrate your data from the source to the destination only by connecting to data sources and executing data integration tasks.

If ROMA Connect cannot access your data source, develop a plug-in to read and write the data source. Besides, the plug-in should serve as a RESTful API for ROMA

Connect to access. Such data source plug-ins are also called connectors, and the data sources accessed through these connectors are custom data sources.

A connector can be directly connected to a data source, which is a data exchange channel between the data source and ROMA Connect. ROMA Connect reads and writes data sources by calling this RESTful API.



After developing and deploying a connector, create it on ROMA Connect and define related parameters for connection.

## Procedure

1. Log in to the ROMA Connect console and choose **Assets** in the navigation pane on the left.
2. Click **Create Connector** in the upper right corner of the page.
3. On the **Create Connector** page, enter configuration information about a connector.

**Table 5-106** Connector parameters

| Parameter   | Description   |
|-------------|---|
| Name        | Connector name. It is recommended that you name it by rule to facilitate search.  |
| Type        | Select the permissions type of the connector for data operations. <ul style="list-style-type: none"><li>• <b>Reader/Writer</b>: reads and writes data sources.</li><li>• <b>Reader</b>: reads data sources only.</li><li>• <b>Writer</b>: writes data sources only.</li></ul> |
| Description | Brief description of the connector.   |

| Parameter            | Description  |
|----------------------|--|
| Data Source Metadata | <p>Parameters for ROMA Connect to access the data source. The message data exchanged between ROMA Connect and the connector is in JSON format, and the parameters are transferred in the key-value format.</p> <p>Click <b>Add Parameter</b> to add the parameters for accessing the data source.</p> <ul style="list-style-type: none"><li>● <b>Parameter Name:</b> name of a parameter to be displayed on the data source configuration page. It is only used to identify a parameter and does not take effect in an actual data integration task.</li><li>● <b>Parameter Key:</b> parameter key transferred in a data source connection request.</li><li>● <b>Parameter Value Type:</b> type of the parameter value to be configured on the data source connection page.<ul style="list-style-type: none"><li>- <b>Text:</b> text box.</li><li>- <b>Select:</b> drop-down list box.</li><li>- <b>Date:</b> date control, which allows you to select a date and time.</li><li>- <b>Textarea:</b> character text box, which supports line breaks.</li></ul></li><li>● <b>Verification Rule:</b> whether to verify the input parameter values.<ul style="list-style-type: none"><li>- <b>None</b></li><li>- <b>No special characters:</b> The system checks whether the parameter value contains special characters. Special characters include digits, letters, hyphens (-), and underscores (_).</li></ul></li><li>● <b>Default Value:</b> default value to be transferred if a parameter is not specified. If this parameter is left blank, no default value will be transferred.</li><li>● <b>Mandatory:</b> whether the parameter is mandatory.</li></ul> |
| Reader Metadata      | <p>Reader parameters of a data source. When the data source is selected as the source of a data integration task, the reader parameters defined here must be set in the data integration task. The message data exchanged between ROMA Connect and the connector is in JSON format, and the parameters are transferred in the key-value format.</p> <p>Click <b>Add Parameter</b> to add the reader parameters of the data source. The parameter configuration description is similar to that of <b>Data Source Metadata</b>.</p>  |

| Parameter       | Description  |
|-----------------|--|
| Writer Metadata | <p>Writer parameters of a data source. When the data source is selected as the destination of a data integration task, the writer parameters defined here must be set in the data integration task. The message data exchanged between ROMA Connect and the connector is in JSON format, and the parameters are transferred in the key-value format.</p> <p>Click <b>Add Parameter</b> to add the writer parameters of the data source. The parameter configuration description is similar to that of <b>Data Source Metadata</b>.</p> |

Figure 5-59 Example of defining connector parameters

| * Data Source Metadata | * Parameter Name | * Parameter Key | Parameter Value Type | Verification Rule | Default Value | Mandatory                           |
|------------------------|------------------|-----------------|----------------------|-------------------|---------------|-------------------------------------|
|                        | host             | host            | Text                 | None              |               | <input checked="" type="checkbox"/> |
|                        | port             | port            | Text                 | None              | 25            | <input checked="" type="checkbox"/> |
|                        | protocol         | protocol        | Text                 | None              | smtp          | <input checked="" type="checkbox"/> |
|                        | userName         | userName        | Text                 | None              |               | <input checked="" type="checkbox"/> |
|                        | password         | password        | Pass...              | None              |               | <input checked="" type="checkbox"/> |
| ⊕ Add Parameter        |                  |                 |                      |                   |               |                                     |
| Writer Metadata        | * Parameter Name | * Parameter Key | Parameter Value Type | Verification Rule | Default Value | Mandatory                           |
|                        | sender           | sender          | Text                 | None              |               | <input checked="" type="checkbox"/> |
| ⊕ Add Parameter        |                  |                 |                      |                   |               |                                     |

4. Click **Create**.

## 5.8.2 Publishing Connectors

### Overview

After creating a connector in ROMA Connect, you need to publish the connector so that ROMA Connect can connect to the deployed connector. ROMA Connect connects to the custom data source by a published connector instance, reading data from and writing data to the data source.

### Prerequisites

- A connector has been developed and deployed offline. For details about how to develop the API for connecting a connector to ROMA Connect, see [RESTful API Specifications of Connectors](#).
- A connector has been created in ROMA Connect, as described in [Creating a Connector](#).

### Procedure

1. Log in to the ROMA Connect console and choose **Assets** in the navigation pane on the left.
2. On the **Connectors** tab page, click **Publish** of the connector.

3. In the dialog box that is displayed, set related parameters and click **OK**.

**Table 5-107** Connector configuration

| Parameter           | Description  |
|---------------------|--|
| Instance Name       | Enter a connector instance name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Connection Address  | Access address of the deployed connector.  |
| Authentication Mode | Select the security authentication mode used for accessing the connector instance. For simplicity, select <b>None</b> . <ul style="list-style-type: none"><li>• <b>None</b>: Access requests are not authenticated.</li><li>• <b>AppKey</b>: Access requests are authenticated using AppKey and AppSecret.</li></ul> |
| AppKey              | Enter the AppKey used for connector authentication.  |
| AppSecret           | Enter the AppSecret used for connector authentication.   |

 **NOTE**

Only connectors in the **Editable** state can be edited. Connectors in the **Published** state cannot be edited.

# 6 Service Integration Guide

---

[Usage Introduction](#)

[Exposing an API](#)

[Exposing a Function API](#)

[Exposing a Data API](#)

[Calling an API](#)

[Managing APIs](#)

[Managing Custom Backends](#)

[Configuring API Control Policies](#)

[Configuring API Plug-in Policies](#)

[Configuring a Custom Authorizer](#)

[Configuring Signature Verification for Backend Services](#)

[Configuring API Cascading](#)

## 6.1 Usage Introduction

### Function Description

APIC is the API integration component of ROMA Connect. It encapsulates APIs, data sources, and custom functions into standard RESTful APIs, and then exposes them to external systems. ROMA Connect has the following advantages for service integration:

- **Convenient API management**  
ROMA Connect provides full-lifecycle management for APIs, including creating, debugging, publishing, taking offline, authorizing, editing, and deleting APIs.
- **Custom API backend services**  
ROMA Connect provides two types of backends.

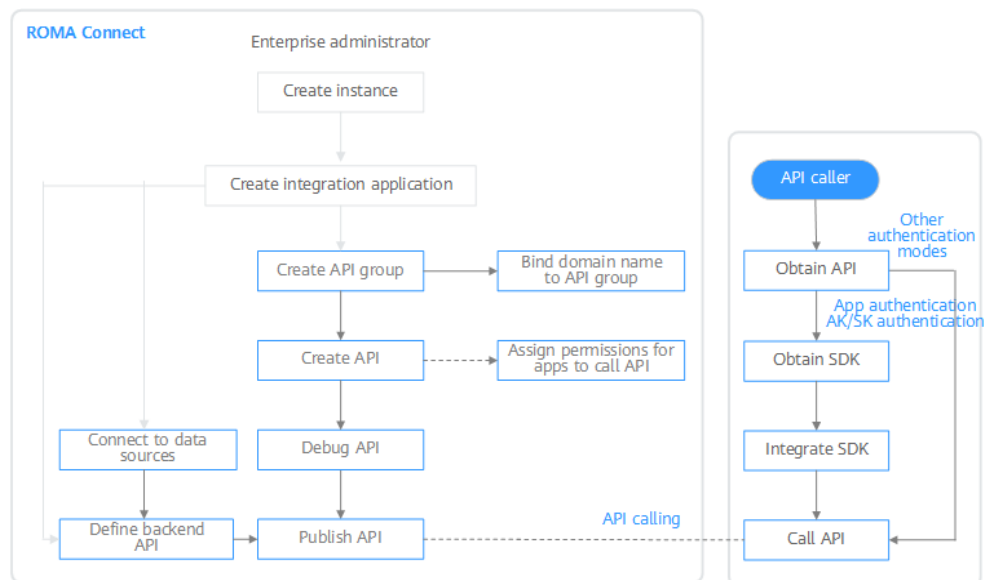


- Data backends, where data sources are exposed as APIs. For details about the supported source types, see [Data Sources Supported by APIC](#).
- Function backends, where function capabilities are exposed as APIs.
- **API monitoring portal**  
ROMA Connect provides a visualized dashboard for API calling and analysis of the performance metrics related to API calling and identifies potential risks that may affect services.
- **Multi-layer security protection**  
ROMA Connect provides multiple authentication modes, refined request throttling, and strict access control for secure API calling.

## Process Flow

The following figure shows how ROMA Connect integrates services.

**Figure 6-1** Process of using ROMA Connect for service integration



1. You have **created an instance and integration application**.
2. **Expose an API.**
  - Exposing APIs
    - i. **Create an API group.**  
Each API belongs to an API group, so create one before creating an API.
    - ii. **Create an API.**  
Encapsulate existing backend services into standard RESTful APIs and expose them to external systems.
    - iii. **Debug the API.**  
Verify that the API service functions are normal using the ROMA Connect debugging function.
    - iv. **Publish the API.**

- Publish an API in an environment so that it can be called.
- v. **Bind a domain name to the API group.**  
Before exposing an API, bind an independent domain name to the API group so that users can access the API.  
(Optional) Before binding an independent domain name to an API, use the default subdomain name to test API calling. ROMA Connect limits the number of times that this default subdomain name can be accessed (1000 times a day).
  - vi. (Optional) **Authorize integration applications to access the API.**  
For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.
- Creating and exposing a data API
- i. **Connect to a data source.**  
Connect to a data source to ensure that data can be read from the data source.
  - ii. **Create a data backend.**  
Define a data source as a backend service to read and write data from the data source.
  - iii. **Publish a data API.**  
Publish the data backend, generate a data API, and expose the API externally.
  - iv. **Bind a domain name to the API group.**  
Before exposing an API, bind an independent domain name to the API group so that users can access the API.  
(Optional) Before binding an independent domain name to an API, use the default subdomain name to test API calling. ROMA Connect limits the number of times that this default subdomain name can be accessed (1000 times a day).
  - v. (Optional) **Authorize integration applications to access the API.**  
For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.
- Creating and exposing a function API
- i. **Create a function backend.**  
Define custom functions as backend services.
  - ii. **Publish a function API.**  
Publish the function backend, generate a function API, and expose the API externally.
  - iii. **Bind a domain name to the API group.**  
Before exposing an API, bind an independent domain name to the API group so that users can access the API.  
(Optional) Before binding an independent domain name to an API, use the default subdomain name to test API calling. ROMA Connect

limits the number of times that this default subdomain name can be accessed (1000 times a day).

- iv. (Optional) **Authorize integration applications to access the API.**

For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.

3. **Call the API.**

Obtain the API and its access address to call the API. This step requires different authentication operations depending on the authentication mode.

## 6.2 Exposing an API

### 6.2.1 Creating an API Group

An API group is a set of APIs the same type of services use. An API developer creates an API group to manage all APIs in the group. Each API belongs to an API group. Before creating an API, create an API group.

#### Constraints

The system allocates a subdomain name to the API group for internal testing. The subdomain name can be accessed up to 1000 times a day. To give your users access to your APIs, bind independent domain names to their API group.

#### Prerequisites

Each API group must belong to an integration application. Before creating an API group, ensure that an integration application is available, or **create one**.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Groups**. In the upper right corner, click **Create API Group** and select **Create Directly**.
3. On the page displayed, configure the following parameters and click **OK**.

**Table 6-1** API group parameters

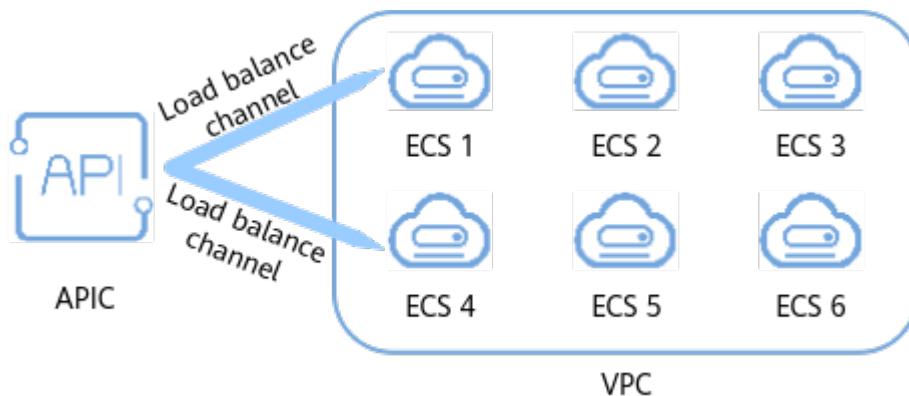
| Parameter | Description  |
|-----------|--|
| Name      | Enter an API group name. Using naming rules facilitates future search. |

| Parameter               | Description  |
|-------------------------|--|
| Scope                   | Specify who can view the API group. <ul style="list-style-type: none"><li>• <b>Integration application:</b> An API group belongs to a specific integration application. Only users who have permissions for this application can view and perform operations on the API group.</li><li>• <b>All:</b> All users in the current instance can view and perform operations on the API group.</li></ul> |
| Integration Application | Mandatory for <b>Scope</b> set to <b>Integration application</b> . Select the API group's integration application. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| Description             | Enter a brief description of the API group.  |

## 6.2.2 (Optional) Creating a Load Balance Channel

This channel allows ROMA Connect to access backend services deployed on servers in load balancing mode (direct access to ECSs in the same VPC, or to ECSs in other VPCs and private servers by specifying IP addresses).

For example, six ECSs deployed in a VPC have a load balance channel to reach ECS 1 and ECS 4. ROMA Connect can access these two ECSs through the channel.



### Prerequisites

- The network between ROMA Connect and the servers in the load balance channel is working.
  - Same VPC: Let the instance directly access the servers.
  - Two VPCs in the same region: Connect the instance and the servers with a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.

- You have the **VPC Administrator** permission.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Load Balance Channels** tab, click **Create Load Balance Channel**.
3. On the page displayed, configure the following parameters.

**Table 6-2** Load balance channel parameters

| Parameter         | Description  |
|-------------------|--|
| Name              | Enter a load balance channel name. Using naming rules facilitates future search.   |
| Port              | Enter the access port number in the load balance channel.  |
| Routing Algorithm | Select an algorithm for routing backend service requests. The channel determines the server to which the requests are to be sent by the algorithm. |

4. Configure servers.
  - a. Select a mode for adding servers to the load balance channel. This mode cannot be changed once the channel is created.
    - **Select cloud servers** from the list.
    - **Specify IP addresses** of the servers.
  - b. Click **Create Server Group**. In the dialog box displayed, configure the following parameters and click **OK**.  
Servers can be added to different groups.

**Table 6-3** Server group parameters

| Parameter   | Description   |
|-------------|---|
| Group Name  | Enter a server group name. Using naming rules facilitates future search.  |
| Weight      | Enter the weight of the server group. The larger weight, the more requests that can be forwarded to the servers in the group. |
| Description | Enter a brief description of the group.   |

- c. Add servers to the load balance channel.
  - Add cloud servers.
    - 1) Click **Add Cloud Server**.

- 2) In the dialog box displayed, select a subnet, select the cloud servers to be added, and click **OK**.
- 3) Configure the following parameters.

**Table 6-4** Cloud server parameters

| Parameter     | Description   |
|---------------|---|
| Standby Node  | After you enable this option, the backend server serves as a standby node. It works only when all non-standby nodes are faulty. |
| Port          | Enter the access port number of the backend server.<br><b>0</b> : uses the port of the load balance channel.                    |
| Server Status | Specify whether to enable the server.<br>When enabled: Requests are distributed to the server.                                  |

- Add a backend server address.
  - 1) Click **Add Backend Server Address**.
  - 2) Configure the following parameters.

**Table 6-5** Backend server parameters

| Parameter              | Description   |
|------------------------|---|
| Backend Server Address | Enter the IP address of the backend server.   |
| Standby Node           | After you enable this option, the backend server serves as a standby node. It works only when all non-standby nodes are faulty. |
| Port                   | Enter the access port number of the backend server.<br><b>0</b> : uses the port of the load balance channel.                    |
| Server Status          | Specify whether to enable the server.<br>When enabled: Requests are distributed to the server.                                  |

5. Configure the health check (enabled by default), or disable this parameter.

**Table 6-6** Health check parameters

| Parameter              | Description  |
|------------------------|--|
| Protocol               | Select the protocol used for the health check.<br>Options: <b>TCP, HTTP, HTTPS</b>   |
| Two-way Authentication | Available for <b>Protocol</b> set to <b>HTTPS</b> .<br>Specify whether to enable two-way authentication between ROMA Connect and backend servers.  |
| Path                   | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Enter the health check URL.  |
| Method                 | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Select the HTTP request method used for the health check.<br>Options: <b>GET</b> or <b>HEAD</b>  |
| Check Port             | Destination port of the health check.<br>Default: uses the port number configured for the load balance channel.  |
| Healthy Threshold      | Number of consecutive successful checks required for an ECS to be considered healthy. Example: If set to <b>2</b> , ROMA Connect declares the ECS status to be healthy when the check is successful twice in a row.    |
| Unhealthy Threshold    | Number of consecutive failed checks required for an ECS to be considered unhealthy. Example: If set to <b>5</b> , ROMA Connect declares the ECS status to be abnormal when the check fails five times in a row.        |
| Timeout (s)            | Response timeout of a health check, in seconds. If no response is received within this time, the health check fails.   |
| Interval (s)           | Interval between consecutive checks, in seconds.   |
| Status Codes           | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>When the server returns a specified HTTP response code, the server considers the response to be successful.<br>Supports multiple response codes. |

6. Click **Finish**.

## 6.2.3 Creating an API

You can create APIs to encapsulate existing backend services into standard RESTful APIs and open up them to other users.

### Prerequisites

- Each API must belong to an integration application. Before creating an API, ensure that an integration application is available, or [create one](#).

- Each API must belong to an API group. Before creating an API, ensure that the API group is available, or [create one](#).
- If you need to use a load balance channel to access the server where the backend service is located, [create a channel](#) first.
- If you need to use custom authentication, [create a frontend custom authorizer](#) first.
- Ensure that the ROMA Connect instance can communicate with your backend service network.
  - Same VPC: Let the instance can directly access the backend service.
  - Two VPCs in the same region: Connect the instance and the backend service with a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.
- To enable cross-VPC private access, [configure the routes](#) between the instance and the backend service subnet.
- To use FunctionGraph as an API backend service, the user must be assigned the **FunctionGraph Administrator** role.
- In the same instance, the group, request method, request path, and matching mode of two APIs cannot be all the same.

## Defining the API Request

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. In the upper right corner, click **Create API**.
3. On the page displayed, configure the frontend definition of the API.

**Table 6-7** Defining the API frontend

| Parameter               | Description  |
|-------------------------|--|
| API Name                | Enter an API name. Using naming rules facilitates future search.   |
| Integration Application | Select an integration application for the API. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| API Group               | Select an API group for the API. You can select only the group under the specified integration application or under an <b>All</b> group. If none is available, click <b>Create API Group</b> on the right to create one.<br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API. |



| Parameter           | Description   |
|---------------------|---|
| URL                 | <p>Configure the API access address.</p> <ul style="list-style-type: none"><li>• <b>Method:</b> Select the API request method.<ul style="list-style-type: none"><li>– <b>ANY:</b> The API is accessible by any request method.</li></ul></li><li>• <b>Protocol:</b> Select the request protocol the API will use. HTTPS is recommended for transmitting important or sensitive data.<ul style="list-style-type: none"><li>– Options: <b>HTTP, HTTPS, HTTP&amp;HTTPS</b></li></ul></li><li>• <b>Path:</b> Enter the request path of the API.<br/>Example: <code>/getUserInfo/{userId}</code>.<br/>The URL can have multiple path parameters, each enclosed by braces.<ul style="list-style-type: none"><li>– Each path parameter must occupy the entire segment between slashes (/).<br/>Wrong example: <code>/abc{userId}</code></li><li>– If <b>Matching</b> is set to <b>Exact match</b>, a plus sign (+) can be added to the end of a path parameter.<br/>Example: <code>/users/{p+}</code>, where <i>p</i> indicates one or multiple segments between slashes (/).</li><li>– <b>Path</b> parameters contained in the request path must be defined as request parameters.</li><li>– <b>Path</b> is case sensitive.</li></ul></li></ul> |
| Matching            | <p>Select the matching mode of the API request path.</p> <ul style="list-style-type: none"><li>• <b>Exact match:</b> The path in an API request must be the same as the value of <b>Path</b>.</li><li>• <b>Prefix match:</b> The path in an API request must be prefixed with the value of <b>Path</b>.<br/>Example: If <b>Path</b> is set to <code>/test/AA</code> and <b>Matching</b> is set to <b>Prefix match</b>, the API can be accessed using <code>/test/AA/BB</code> or <code>/test/AA/CC</code> but cannot be accessed using <code>/test/AACC</code>.</li></ul> <p><b>NOTE</b><br/>If you set the matching mode to <b>Prefix match</b>, the characters of the API request path excluding the prefix are transparently transmitted to the backend service.<br/>Example: If the request path is set to <code>/test</code> and the backend request path is set to <code>/test2</code>, the request path received by the backend service is <code>/test2/AA/CC</code> when <code>/test/AA/CC</code> is used to access the API.</p>  |
| Tags                | Add tags for the API for quick search.  |
| Description         | Enter a brief description of the API.   |
| Content Format Type | Available for <b>Method</b> set to <b>POST, PUT, or ANY</b> .<br>Determine whether to specify the content format of API requests. The options are <b>application/json, application/xml, text/plain</b> , and <b>multipart/form-data</b> .   |

| Parameter | Description  |
|-----------|--|
| Body      | Available for <b>Method</b> set to <b>POST</b> , <b>PUT</b> , <b>PATCH</b> , or <b>ANY</b> .<br>Enter the description of the request body in the API request to help the API caller understand how to correctly encapsulate the API request. |

4. Configure API security information.

**Table 6-8** Security configuration

| Parameter                 | Description  |
|---------------------------|--|
| Visibility                | Determine whether the API can be listed on KooGallery. <ul style="list-style-type: none"><li>• <b>Public</b>: The API can be listed on KooGallery.</li><li>• <b>Private</b>: The API cannot be listed on KooGallery even when its API group has been listed.</li></ul>   |
| Authentication Mode       | Select the authentication mode of the API. The App authentication mode is recommended. <ul style="list-style-type: none"><li>• <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an authorized integration application. APIs using this mode can be called by all users.</li><li>• <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK. APIs using this mode can be called only by users on the same cloud service platform.</li><li>• <b>Custom</b>: The custom function API is used for authenticating API requests. APIs using this mode can be called by all users.</li><li>• <b>None</b>: Authentication is not required for API requests. APIs using this mode can be called by all users.</li></ul> |
| Simple Authentication     | Available for <b>Authentication Mode</b> set to <b>App</b> .<br>This parameter specifies whether to use simple security authentication for API calling. It takes effect only when the API request protocol is HTTPS. Once enabled, a user gets authenticated using the AppCode when calling an API. Signature verification is not required for API requests.   |
| Two-Factor Authentication | Available for <b>Authentication Mode</b> set to <b>App</b> or <b>IAM</b> .<br>This parameter specifies whether to use a custom function API to authenticate API requests when App or IAM authentication is also enabled.<br>Once enabled, select a frontend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a frontend custom authorizer.   |

| Parameter         | Description   |
|-------------------|---|
| Custom Authorizer | Mandatory for <b>Authentication Mode</b> set to <b>Custom</b> .<br>Select a frontend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a frontend custom authorizer.   |
| CORS              | Specifies whether cross-origin resource sharing (CORS) is supported for the API.<br>For security, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send <b>XMLHttpRequest</b> requests to a server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a> . |

5. Configure the request parameters of the API.

Click **Add Request Parameter** on the **Parameters**, **Headers**, and **Cookies** tab pages to configure API request parameters. Path variables also need to be configured if the API's URL contains path parameters.

**Table 6-9** Request parameters

| Parameter          | Description   |
|--------------------|---|
| Parameter Name     | Name of the request parameter.  |
| Parameter Type     | Data type of the request parameter.<br>Options: <b>STRING</b> or <b>NUMBER</b>  |
| Required           | Whether the request parameter must be specified when the API is called.<br><b>No</b> : enables you to set <b>Default Value</b> .  |
| Passthrough        | Whether to transparently transmit the parameter to the backend service.   |
| Enumerated Value   | Enumerated value of the parameter. Use commas (,) to separate multiple enumerated values. The value of this parameter can only be one of the enumerated values.   |
| Default Value      | Available for <b>Required</b> set to <b>No</b> .  |
| Value Restrictions | If <b>Parameter Type</b> is set to <b>STRING</b> , set the minimum and maximum lengths of the parameter value. If <b>Parameter Type</b> is set to <b>NUMBER</b> , set the minimum and maximum parameter values.<br>Setting both <b>Min. Length/Min. Value</b> and <b>Max. Length/Max. Value</b> to <b>0</b> indicates no limit. |

| Parameter   | Description                           |
|-------------|---------------------------------------|
| Example     | Example of a request parameter value. |
| Description | Description of the request parameter. |

 **NOTE**

- The parameter name cannot be **x-stage** or start with **x-apig-** or **x-sdk-** (case insensitive).
  - For headers, the name is case-insensitive and must be unique. Do not use **X-Auth-Token** or **Authorization** for IAM authentication, or **Authorization** for app authentication.
6. Click **Next**.

## Defining the Backend

1. Set the backend type to **HTTP&HTTPS**, **FunctionGraph**, or **Mock**.

 **NOTE**

- If FunctionGraph is not deployed in the current environment, the backend type cannot be set to **FunctionGraph**.
2. Configure the basic definition of the default backend. The configuration varies by the backend type selected.
    - Default backend of the HTTP&HTTPS type

**Table 6-10** Default backend of the HTTP&HTTPS type

| Parameter            | Description  |
|----------------------|--|
| Load Balance Channel | Determine whether to use a load balance channel to access backend services.<br><b>Configure:</b> <a href="#">create a load balance channel</a> in advance. |

| Parameter | Description  |
|-----------|--|
| URL       | <p>Configure the URL of the backend service.</p> <ul style="list-style-type: none"><li>● <b>Method:</b> Select the request method of the backend service. <b>ANY</b> indicates that the backend service can be accessed using any request method.</li><li>● <b>Protocol:</b> Select the request protocol used by the backend service. WebSocket is supported. <b>HTTPS</b> is recommended for transmitting important or sensitive data.</li><li>● <b>Backend Address:</b> Mandatory for <b>Load Balance Channel</b> set to <b>Skip</b>. Enter the access address of the backend service in the format of Host:Port.<br/><i>Host:</i> IP address or domain name for accessing the backend service. If no port is specified, port 80 and 443 is used by default for HTTP and HTTPS respectively.<ul style="list-style-type: none"><li>– If the backend address needs to contain environment variables, use <i>#Variable name#</i> to add the environment variables to the backend address, for example, <b>#ipaddress#</b>. Multiple environment variables can be added, for example, <b>#ipaddress##test#</b>.</li><li>– If you want to call a custom backend by using its backend request address, add two built-in gateway parameters to <b>system parameters</b>.<br/><b>appid:</b> Set <b>Backend Parameter Name</b> to <b>x-auth-app</b> and <b>Backend Parameter Location</b> to <b>HEADER</b>.<br/><b>providerAppId:</b> Set <b>Backend Parameter Name</b> to <b>x-ld-appid</b> and <b>Backend Parameter Location</b> to <b>HEADER</b>.</li></ul></li><li>● <b>Load Balance Channel:</b> Mandatory for <b>Load Balance Channel</b> set to <b>Configure</b>. Select a load balance channel for backend service access.</li><li>● <b>Path:</b> Enter the request path of the backend service. Enclose multiple path parameters by braces. Example: <b>/getUserInfo/{userId}</b><br/>If the path needs to contain an environment variable, enclose the environment variable in number signs (#).<br/>Example: <b>/#path#</b><br/>Environment variable names are case sensitive. Multiple environment variables can be added.<br/>Example: <b>/#path##request#</b></li></ul> |

| Parameter              | Description  |
|------------------------|--|
|                        | <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>If you have defined environment variables in <b>Path</b>, the API cannot be debugged on the API debugging page.</li> <li>For variables defined in <b>Path</b>, corresponding environment variables and their values must be configured. Otherwise, the API cannot be published because there will be no values that can be assigned to the variables.</li> </ul>   |
| Cascading              | <p>Available for the <b>instance parameter cascade</b> set to <b>on</b> and <b>Load Balance Channel</b> set to <b>Configure</b>.</p> <p>Whether to use the cascading mode to access backend services. This function must be enabled when <b>API cascading</b> is used.</p>   |
| Host Header            | <p>Available for <b>Load Balance Channel</b> set to <b>Configure</b>.</p> <p>Define the Host header field carried in the backend service request.</p>  |
| Timeout (ms)           | <p>Enter the timeout interval of a backend service request.</p> <p>Default: <b>5000</b></p>  |
| Retries                | <p>Number of retries after ROMA Connect fails to call the backend service.</p> <ul style="list-style-type: none"> <li><b>-1</b>: The retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li> <li><b>0 to 10</b>: This parameter is enabled and will make the configured number of retries.</li> </ul> <p><b>NOTE</b><br/>If <b>Load Balance Channel</b> is set to <b>Configure</b>, the number of retries must be less than the number of enabled backend servers in the load balance channel.</p> |
| Two-Way Authentication | <p>Available only if the URL's <b>Protocol</b> is set to <b>HTTPS</b>.</p> <p>Determine whether to enable two-way authentication between ROMA Connect and backend services. If you enable this option, you also need to configure the certificate for client authentication.</p>   |
| Backend Authentication | <p>Specify whether to use a custom authorizer for authentication.</p> <p>Once enabled, select a backend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer.</p>  |

- Default backend of the FunctionGraph type

**Table 6-11** Default backend of the FunctionGraph type

| Parameter              | Description   |
|------------------------|---|
| Function URN           | Enter the ID of a function request. Click <b>Select</b> to add a backend function URN.  |
| Version/Alias          | Select the version or alias of the function to be used.   |
| Invocation Mode        | Select the invocation type of the function. <ul style="list-style-type: none"><li>• <b>Synchronous</b>: When receiving a calling request, FunctionGraph immediately processes the request and returns a result. The client closes the connection once it has received a response from the backend.</li><li>• <b>Asynchronous</b>: After receiving a calling request, FunctionGraph queues the request and returns the result after the request is successfully executed. The server processes the queuing requests one by one when it is idle. The client does not care about the calling result.</li></ul> |
| Timeout (ms)           | Enter the timeout interval of a backend service request. The default value is <b>5000</b> .   |
| Backend Authentication | Specify whether to use a custom authorizer for authentication.<br>Once enabled, select a backend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer.  |

- Default backend of the Mock type

**Table 6-12** Default backend of the Mock type

| Parameter              | Description  |
|------------------------|--|
| Status Code            | Select the HTTP status code returned by the API.   |
| Response               | Enter the API response result.<br>Example: Set <b>Response</b> to <b>Successful Info</b> , so the API always returns <b>Successful Info</b> .  |
| Backend Authentication | Specify whether to use a custom authorizer for authentication.<br>Once enabled, select a backend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer. |
| Add Header             | Click <b>Add Header</b> to add custom header parameters of the API response.   |

3. (Optional) Configure backend parameters to map the request parameters transferred when the API is called to the corresponding location in the backend request. If no request parameter is defined in the API request, skip this step.
  - a. In the **Backend Parameters** area, add backend parameters in either of the following ways:
    - Click **Import Request Parameter** to add all defined API request parameters to the backend parameters.
    - Click **Add Backend Parameter Mapping** to add backend parameters one by one as required.
  - b. Modify the mapping between API request parameters and backend parameters.
    - The name and location of the backend parameters can be different from those of the input parameters.
    - If **Backend Parameter Location** is **PATH**, the backend parameter name must be the same as the parameter name configured in **Path** in the backend URL.
    - The backend parameter name cannot be **x-stage** or start with **x-apig-** or **x-sdk-** (case insensitive).
    - If **Backend Parameter Location** is **HEADER**, the parameter name is case insensitive.

The parameters and backend path in the following table are used as an example. Parameters **test01** and **test03** are located in **PATH** and **QUERY** in an API request. Through parameter mappings, the backend service receives the values of **test01** and **test03** in **HEADER**. Parameter **test02** is located in **HEADER** in an API request. Through the parameter mapping, the backend service receives the value of **test02** using **test05** in **PATH**.

**Table 6-13** Example of backend parameter mapping

| Request Parameter Name | Request Parameter Location | Backend Parameter Name | Backend Parameter Location |
|------------------------|----------------------------|------------------------|----------------------------|
| test01                 | PATH                       | test01                 | HEADER                     |
| test02                 | HEADER                     | test05                 | PATH                       |
| test03                 | QUERY                      | test03                 | HEADER                     |

Backend path: `/apitest/{test05}`

Assume that **test01** is **aaa**, **test02** is **bbb**, and **test03** is **ccc**.

The API request is as follows:

```
curl -ik -H 'test02:bbb' -X GET https://example.com/v1.0/aaa?test03=ccc
```



The backend service request is as follows:

```
curl -ik -H 'test01:aaa' -H 'test03:ccc' -X GET https://apitest/bbb
```

- (Optional) Configure backend constant parameters. Constant parameters can be defined to receive fixed constants. When sending a request to a backend service, ROMA Connect adds constant parameters to the specified location in the request and then sends the request to the backend service.

In the **Constant Parameters** area, click **Add Constant Parameter** to add the constant parameters of the backend service request.

**Table 6-14** Constant parameters

| Parameter               | Description  |
|-------------------------|--|
| Constant Parameter Name | Enter the name of a constant parameter. If <b>Parameter Location</b> is <b>PATH</b> , the parameter name must be the same as that in the backend URL.<br><br><b>NOTE</b> <ul style="list-style-type: none"> <li>The parameter name cannot be <b>x-stage</b> or start with <b>x-apig-</b> or <b>x-sdk-</b> (case insensitive).</li> <li>If <b>Location</b> is <b>HEADER</b>, the parameter name is case insensitive.</li> </ul> |
| Parameter Location      | Select the location of the constant parameter in the backend service request.<br><br>Options: <b>PATH, HEADER, QUERY</b>   |
| Parameter Value         | Enter the value of the constant parameter.   |
| Description             | Enter the description of the constant parameter.   |

#### NOTE


- ROMA Connect sends requests containing constant parameters to backend services after percent-encoding of special parameter values. Ensure that the backend services support percent-encoding. For example, parameter value **[api]** becomes **%5Bapi%5D** after percent-encoding.
  - For values of path parameters, ROMA Connect will percent-encode the following characters: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters `?></%#"[ \]^`{ }`
  - For values of query parameters, ROMA Connect will percent-encode the following characters: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters `>=<+&%#"[ \]^`{ }`
- (Optional) Configure backend system parameters. If a backend service needs to receive parameter information generated during system running, such as gateway built-in parameters, frontend authentication parameters, and backend authentication parameters, you can set system parameters. When sending a request to a backend service, ROMA Connect adds system parameters to the specified location in the request and then sends the request to the backend service.

In the **System Parameters** area, click **Add System Parameter** to add the system parameters of the backend service request.

**Table 6-15** System parameters

| Parameter             | Description   |
|-----------------------|---|
| System Parameter Type | <p>Select the type of a system parameter.</p> <ul style="list-style-type: none"><li>● <b>Default gateway parameter:</b> system parameters that can be configured for ROMA Connect.</li><li>● <b>Frontend authentication parameter:</b> parameters to be displayed in the frontend custom authentication result. This option is available only if you have set <b>Authentication Mode</b> to <b>Custom</b> in <a href="#">Defining the API Request</a>.</li><li>● <b>Backend authentication parameter:</b> parameters to be displayed in the backend custom authentication result. This option is available only if <b>Backend Authentication</b> is enabled when you <a href="#">configure the basic definition of the default backend</a>.</li></ul>   |
| System Parameter Name | <p>Enter the name of a system parameter.</p> <ul style="list-style-type: none"><li>● If <b>System Parameter Type</b> is <b>Default gateway parameter</b>, select the parameters that can be obtained by the system.<ul style="list-style-type: none"><li>– <b>sourceIp:</b> source IP address of the client that calls an API.</li><li>– <b>stage:</b> name of the environment in which the API is published.</li><li>– <b>apiId:</b> ID of the API.</li><li>– <b>appId:</b> ID of the integration application used to call the API.</li><li>– <b>requestId:</b> request ID generated when the API is called.</li><li>– <b>serverAddr:</b> IP address of the gateway server.</li><li>– <b>serverName:</b> name of the gateway server.</li><li>– <b>handleTime:</b> processing time of the called API.</li><li>– <b>providerAppId:</b> ID of the integrated application to which the API belongs.</li><li>– <b>apiName:</b> name of the API. This parameter is available only after the API is published.</li><li>– <b>appName:</b> name of the integration application used to call the API.</li></ul></li><li>● If <b>System Parameter Type</b> is set to <b>Frontend authentication parameter</b> or <b>Backend authentication parameter</b>, you can define a value for <b>System Parameter Name</b>. However, the customized value must be set in the return result of custom authentication.</li></ul> |

| Parameter                  | Description  |
|----------------------------|--|
| Backend Parameter Name     | Enter the name of the backend parameter to be mapped.<br><b>NOTE</b> <ul style="list-style-type: none"> <li>The parameter name cannot be <b>x-stage</b> or start with <b>x-apig-</b> or <b>x-sdk-</b> (case insensitive).</li> <li>If <b>Backend Parameter Location</b> is <b>HEADER</b>, the parameter name is case insensitive.</li> </ul> |
| Backend Parameter Location | Select the location of the system parameter in the backend service request.<br>Options: <b>PATH, HEADER, QUERY</b>   |
| Description                | Enter the description of the system parameter.   |

6. (Optional) Add a backend policy. You can add multiple backend policies for an API as required and set different policy conditions to forward API requests to different backend services.
  - a. Click  next to **Backend Policies** to add a backend policy for the API.
  - b. Configure information about the backend policy.

Some basic parameters of a backend policy are the same as those of the default backend. For details, see the parameter description for the [default backend basic definition](#). [Table 6-16](#) describes only the parameters specific to a backend policy.

**Table 6-16** Parameters for creating a backend policy

| Parameter      | Description   |
|----------------|---|
| Name           | Enter a backend policy name to identify different backend policies.   |
| Effective Mode | Select the effective mode of the backend policy. <ul style="list-style-type: none"> <li><b>Any condition met:</b> If an API request meets any of the conditions in a policy, the request is forwarded to the backend.</li> <li><b>All conditions met:</b> API requests are forwarded to the backend only when all policy conditions are met.</li> </ul> |

| Parameter         | Description   |
|-------------------|---|
| Policy Conditions | <p>Click <b>Add Policy Condition</b> to add conditions for the policy to take effect.</p> <ul style="list-style-type: none"> <li>● <b>Source:</b> source of the conditions specified in the policy <ul style="list-style-type: none"> <li>– <b>Request parameter:</b> The request parameters set as policy conditions must have already been defined for the API.</li> <li>– <b>Source IP address:</b> IP address from which the API is called</li> <li>– <b>System parameter:</b> a system parameter that defines system runtime for the API</li> <li>– <b>Cookie:</b> cookies of an API request</li> </ul> </li> <li>● <b>Parameter Name:</b> mandatory only if <b>Source</b> is set to <b>Request parameter</b>, <b>System parameter</b>, or <b>Cookie</b>. <ul style="list-style-type: none"> <li>– When you set <b>Source</b> to <b>Request parameter</b>, select a request parameter that has been defined.</li> <li>– When you set <b>Source</b> to <b>Request parameter</b>, select a system parameter. <ul style="list-style-type: none"> <li>- <b>reqPath:</b> request URL<br/>Example: <b>/a/b/c</b></li> <li>- <b>reqMethod:</b> request method<br/>Example: <b>GET</b></li> </ul> </li> <li>– When you set <b>Source</b> to <b>Cookie</b>, enter the name of a cookie parameter.</li> </ul> </li> <li>● <b>Parameter Location:</b> available only if you set <b>Source</b> to <b>Request parameter</b>.</li> <li>● <b>Condition Type:</b> mandatory only if you set <b>Source</b> to <b>Request parameter</b>, <b>Cookie</b>, or <b>System parameter</b>. <ul style="list-style-type: none"> <li>– <b>Equal:</b> The request parameter must be equal to the specified value.</li> <li>– <b>Enumerated:</b> The request parameter must be equal to any of the enumerated values.</li> <li>– <b>Matching:</b> The request parameter must match the regular expression.</li> </ul> </li> </ul> <p><b>NOTE</b><br/>When <b>Source</b> is set to <b>System parameter</b> and <b>Parameter Name</b> to <b>reqMethod</b>, you can set the condition type only to <b>Equal</b> or <b>Enumerated</b>.</p> <ul style="list-style-type: none"> <li>● <b>Condition Value:</b> Enter the value of the judgment condition. <ul style="list-style-type: none"> <li>– If <b>Condition Type</b> is set to <b>Equal</b>, enter a value.</li> </ul> </li> </ul> |

| Parameter | Description   |
|-----------|---|
|           | <ul style="list-style-type: none"><li>– If <b>Condition Type</b> is set to <b>Enumerated</b>, enter multiple values and separate them with commas (,).</li><li>– If <b>Condition Type</b> is set to <b>Matching</b>, enter a regular expression, for example, [0-5].</li><li>– If <b>Source</b> is set to <b>Source IP address</b>, enter one or more IP addresses and separate them with commas (,).</li></ul> |

For example, there are three policy conditions whose **Source** is **Request parameter**, as listed in [Table 6-17](#). If the request parameter value is **11**, policy A is met. If the request parameter value is **5**, policy B is met. If the request parameter value is **15**, policy C is met.

**Table 6-17** Policy parameters

| Policy   | Condition Type | Condition Value |
|----------|----------------|-----------------|
| Policy A | Equal          | 11              |
| Policy B | Enumerated     | 1, 2, 5, 8      |
| Policy C | Matching       | [13-20]         |

7. Configure response examples to help API callers understand the responses to an API request.

**Table 6-18** Response configuration

| Parameter                | Description  |
|--------------------------|--|
| Example Success Response | Example of a successful response returned when the API is successfully called. |
| Example Failure Response | Example of a failure response returned when the API fails to be called.        |

8. Click **Finish**.

## 6.2.4 Debugging an API

Debug APIs to ensure that they function properly.

### Constraints

- APIs whose backend request paths contain environment variables cannot be debugged.
- During API debugging, the request throttling policy bound to the API becomes invalid.

- If the value of **Timeout (ms)** multiplied by the value of **Retries** is greater than 30 seconds in [Defining the Backend](#), API debugging will time out.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. Then choose **More > Debug** of an API.
3. On the **APIs** tab displayed, click **Debug** in the upper right corner and configure the request parameters.

**Table 6-19** Request parameter configuration

| Parameter  | Description  |
|------------|--|
| Parameters | <ul style="list-style-type: none"><li>• <b>Path Variables:</b> Path variables can be configured only if the API request path contains path parameters.</li><li>• <b>Query Strings:</b> Click <b>Add Request Parameter</b> to add and configure query parameters.</li></ul> |
| Headers    | Click <b>Add Request Parameter</b> to add and configure header parameters.   |
| Body       | Add body parameters to the edit box. TEXT, JSON, and XML formats are supported.  |

4. When the configuration is complete, click **Debug**. The request sent when you call the API and the response received are displayed in the lower part of the page.
  - Status code 200 and a normal response body: API called successfully
  - Status code 4xx or 5xx and error code description: API call failed. For details, see [Appendix: API Error Codes](#).

You can send more requests with different parameters and values to verify the API.

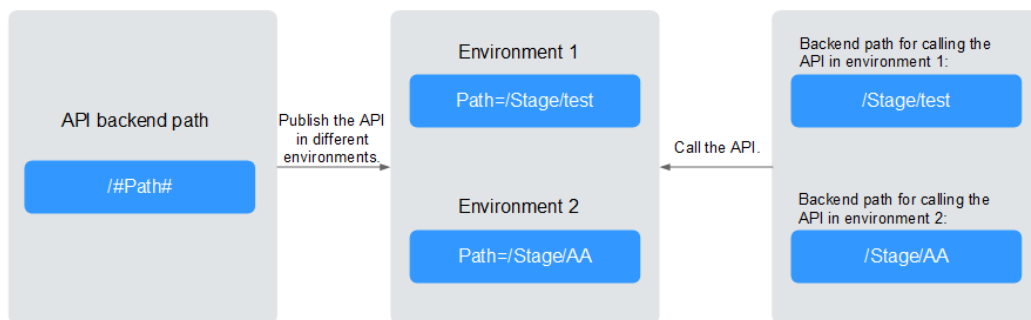
## 6.2.5 (Optional) Creating an Environment and Variable

An API can be called only after it is published in an environment. Publish it in different custom environments for development and testing. **RELEASE** is the system default environment for publishing APIs.

Environment variables are specific to environments. If environment variables are defined in backend information of an API, you need to add the variables to the corresponding environment. You can add variables in different environments to call different backend services using the same API.

Example: an API whose variable **Path** is defined in the backend request path. If variable **Path** is configured for environment 1 and is assigned value **/Stage/test**, use the backend request path **/Stage/test** to call this API in this environment. If variable **Path** is configured for environment 2 and is assigned value **/Stage/AA**, use the backend request path **/Stage/AA** to call this API in this environment.

**Figure 6-2** Application of environment variables



## Creating an Environment

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Environments** tab, click **Create Environment**.
3. In the dialog box displayed, configure environment information and click **OK**.

When a user needs to call an open API, APIs in the **RELEASE** environment will be called by default. To access an API in another environment, add the **X-Stage** parameter to the header of the API request, where the parameter is the environment name. For example, to access an API in the Develop environment, add **X-Stage: Develop** to the header of the API request.

**Table 6-20** Parameters for creating an environment

| Parameter   | Description  |
|-------------|--|
| Name        | Enter an environment name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Description | Enter a brief description of the environment.  |

## Creating an Environment Variable

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Groups**. Click the name of an API group to view details.
3. In the **Environment Variables** area on the **Group Information** tab, select the environment to which you want to add variables, and click **Add Environment Variable**.
4. In the dialog box displayed, configure environment variable information and click **OK**.

**Table 6-21** Parameters for creating an environment variable

| Parameter | Description   |
|-----------|---|
| Name      | Enter the variable name, which must be the same as that defined in the API backend service information. |
| Value     | Enter the value of the environment variable.  |

## 6.2.6 Publishing an API

After creating an API, publish it in an environment so that it can be called by other users.

### Constraints

- If a published API is modified, you must publish it again for the modifications to take effect in the environment.
- If the API has already been published in an environment, publishing it again will overwrite the existing one.

### Prerequisites

**RELEASE** is the system default environment. To publish APIs in another environment, [create an environment](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect** > **APIs**. Then choose **Publish** of an API.
3. On the **Publish APIs** panel, configure publishing information and click **OK**.

**Table 6-22** Parameters for publishing an API

| Parameter   | Description   |
|-------------|---|
| Environment | Select the environment to publish the API in. Or click <b>Create Environment</b> to create one. |
| Description | Enter the description of the API publication.   |

After the API is published, click the API name and go to the **APIs** tab. Click **More** in the upper right corner and choose **View Publishing Records** for details. You can also view the API configuration of each released version and switch to a historical version.



## 6.2.7 Binding Domain Names

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are private or public.

- Private domain name: Service systems deployed on the cloud service platform use these names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform use these names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  - a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  - a. Apply for a public zone from Domain Registration.
  - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

### Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Groups**. Click the name of an API group to view details.
3. On the **Group Information** tab, click **Bind Independent Domain Name**.
4. In the displayed dialog box, enter the domain name and click **OK**.

**Table 6-23** Independent domain name configuration

| Parameter   | Description                    |
|-------------|--------------------------------|
| Domain Name | Enter the domain name to bind. |

| Parameter                      | Description  |
|--------------------------------|--|
| Minimum TLS Version            | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected.   |
| HTTP-to-HTTPS Auto Redirection | Whether to support HTTP-to-HTTPS redirection. Redirection takes effect only when the API request protocol is HTTPS or HTTP&HTTPS and an SSL certificate has been bound to the independent domain name.<br><b>NOTE</b><br>Redirection is only suitable for GET and HEAD requests. Redirecting other requests may cause data loss due to browser restrictions. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate to the independent domain name.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.
    - If a CA certificate has been configured for the SSL certificate, client authentication (HTTPS two-way authentication) is enabled by default.
    - If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

## 6.2.8 (Optional) Authorizing Credentials to Call APIs

Credentials need to be authorized to call APIs that use App authentication. When calling an API, a user is authenticated using the key and secret of a credential.

### NOTE

An API's integration application can directly call the API.

## Constraints

The security authentication mode of the API is set to **App**.

## Prerequisites

The API has been published in an environment. Otherwise, [publish the API](#) first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. Choose **More > Authorize Credentials** of an API.

3. On the page displayed, click **Select Credentials**.
4. Configure authorization information and click **OK**.  
After the authorization is complete, a list of authorized credentials will be displayed.

**Table 6-24** Authorization configuration

| Parameter         | Description  |
|-------------------|--|
| Environment       | Select the environment the API has been published in.  |
| Credentials       | Select the credentials you want to authorize.  |
| Access Parameters | Set access parameters for the selected credentials to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel     | Enabling Green Channel allows whitelisted clients to call the API without authentication.  |
| Whitelist         | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. Whitelisted clients can call the API without authentication.   |
| Blacklist         | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. Blacklisted clients are not allowed to call the API.   |

## 6.3 Exposing a Function API

### 6.3.1 Creating a Function Backend

Define custom functions as backend services so that they can be accessed externally via APIs.

#### Prerequisites

[Create a signature key](#) for any backend services that need one.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, click **Create Backend**.

3. On the **Create Backend** page, set backend parameters and click **Create**.  
After the backend is created, the online IDE of the backend is automatically displayed. The backend type defaults to data backend.

**Table 6-25** Backend configuration

| Parameter                       | Description   |
|---------------------------------|---|
| Name                            | Enter a backend name. Using naming rules facilitates future search.   |
| Integration Application         | Select an integration application for the backend. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Backend Request Method          | Options: <b>GET, POST, PUT, DELETE</b>  |
| Backend Request Path            | The path is case sensitive.<br>Example: <i>/getUserInfo/userId</i>  |
| Backend Security Authentication | Options: <ul style="list-style-type: none"><li>● <b>Signature Key</b>: Authenticate requests using a signature key. This key must bind the frontend API.</li><li>● <b>None</b>: No authentication is required for requests.</li></ul>   |
| Description                     | Enter a description of the backend.   |
| Advanced Settings               |   |
| Version                         | Enter the backend version to differentiate backend services.<br>Example: <b>V1.0</b>  |
| Input Parameters                | Define request parameters of the custom backend service in the <b>Input Parameters</b> area. Click <b>Add Input Parameter</b> and add: <ul style="list-style-type: none"><li>● <b>Name</b>: request parameter name</li><li>● <b>Parameter Location</b>: location of the request parameter in the backend request.<br/>Options: <b>Headers</b> or <b>Parameters</b></li><li>● <b>Default Value</b>: used only in the subsequent custom backend test procedure. This parameter does not take effect during custom backend deployment.</li><li>● <b>Mandatory</b>: whether a request parameter is mandatory in a backend request.</li><li>● <b>Description</b>: description of the request parameter</li></ul> |
| Returned Type                   | Select the response data format of the backend.<br>Options: <b>JSON, XML, STREAM</b>  |

4. Configure the function backend.
  - a. Choose **File > New Function Backend > Blank Template** from the menu bar. In the dialog box displayed, click **Yes** to switch the default backend type to a function backend.
  - b. Compile the function script on the right pane. Alternatively, edit sample scripts provided by the system.  
For details, see [Developing Custom Function Backends](#)

 **NOTE**

The maximum function API script size is 32 KB.

- c. Click **Save** in the upper right corner.
5. Test the backend functions.

In the upper right corner, click **Test**. Add request parameters to the **Test Parameters** area to match the backend definition and click **Test** to send the request.

  - Under **Execution Result**, view the backend response.
  - Under **Execution History**, view the historical test records of the backend. Click a test record to import historical test parameters to the test parameter list on the left for reruns.
6. Deploy the backend.

After the backend is tested, click **Deploy** in the upper right corner. In the dialog box displayed, click **Yes** to deploy the backend.

## 6.3.2 Publishing a Function API

After creating a function backend, publish it as an API in an environment so it can be called by other users.

### Prerequisites

- Each API must belong to an API group. Before publishing a function API, ensure that there are available API groups. Otherwise, [create an API group](#) first.
- If you need to use custom authentication, [create a frontend custom authorizer](#) first.

### Constraints

If request parameters are added to the request path of a frontend API published by a custom backend, the API cannot be published on the publish page of the custom backend. You need to publish the API on the frontend API page. If just the content of the custom backend is modified, deploy the backend again and no publishing is required.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**.

3. In the backend list, click the name of a custom backend. The **Online IDE** page will be displayed.
4. Click **Publish** in the upper right corner.
5. On the page displayed, configure API information and click **Publish** to create a frontend function API for the backend and publish the API in an environment.

**Table 6-26** Parameters for publishing an API

| Parameter                        | Description   |
|----------------------------------|---|
| API Group                        | Select an API group for the frontend API. If none is available, click <b>Create API Group</b> on the right to create one.<br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API.   |
| Environment                      | Select an environment to publish the API in. Or click <b>Create Environment</b> to create one.  |
| Frontend Security Authentication | Select the authentication mode of the API. The App authentication mode is recommended. <ul style="list-style-type: none"><li>● <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an integration application.</li><li>● <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK.</li><li>● <b>Custom</b>: The custom function API is used for authenticating API requests.</li><li>● <b>None</b>: Authentication is not required for API requests.</li></ul> |
| Custom Authorizer                | Mandatory for <b>Frontend Security Authentication</b> set to <b>Custom</b> .<br>Select a frontend custom authorizer you have created.   |
| Frontend Request Protocol        | Select the request protocol used by the frontend API. HTTPS is recommended for transmitting important or sensitive data.<br>Options: <b>HTTP, HTTPS, HTTP&amp;HTTPS</b>   |
| Timeout (ms)                     | Enter the timeout interval of a backend service request.<br>Default: <b>60000</b>   |

| Parameter               | Description   |
|-------------------------|---|
| Retries                 | Number of retries after ROMA Connect fails to call the backend service. <ul style="list-style-type: none"><li>• -1: The retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>• 0 to 10: This parameter is enabled and will make the configured number of retries.</li></ul>                   |
| Advanced Settings       |   |
| Frontend Request Method | Select the request method of the frontend API.<br><b>ANY</b> : The API is accessible by any request method.   |
| Frontend Request Path   | The path is case sensitive.<br>Example: <code>/getUserInfo/userId</code>  |
| CORS                    | Specifies whether CORS is supported for the API.<br>For security, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send <b>XMLHttpRequest</b> requests to a server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a> . |

### 6.3.3 Binding Domain Names

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are private or public.

- Private domain name: Service systems deployed on the cloud service platform use these names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform use these names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

#### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  - a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  - a. Apply for a public zone from Domain Registration.
  - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

## Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Groups**. Click the name of an API group to view details.
3. On the **Group Information** tab, click **Bind Independent Domain Name**.
4. In the displayed dialog box, enter the domain name and click **OK**.

**Table 6-27** Independent domain name configuration

| Parameter                      | Description  |
|--------------------------------|--|
| Domain Name                    | Enter the domain name to bind.   |
| Minimum TLS Version            | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected.   |
| HTTP-to-HTTPS Auto Redirection | Whether to support HTTP-to-HTTPS redirection. Redirection takes effect only when the API request protocol is HTTPS or HTTP&HTTPS and an SSL certificate has been bound to the independent domain name.<br><b>NOTE</b><br>Redirection is only suitable for GET and HEAD requests. Redirecting other requests may cause data loss due to browser restrictions. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate to the independent domain name.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.
    - If a CA certificate has been configured for the SSL certificate, client authentication (HTTPS two-way authentication) is enabled by default.



- If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

## 6.3.4 (Optional) Authorizing Credentials to Call APIs

Credentials need to be authorized to call APIs that use App authentication. When calling an API, a user is authenticated using the key and secret of a credential.

### NOTE

An API's integration application can directly call the API.

## Constraints

The security authentication mode of the API is set to **App**.

## Prerequisites

The API has been published in an environment. Otherwise, [publish the API](#) first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. Choose **More > Authorize Credentials** of an API.
3. On the page displayed, click **Select Credentials**.
4. Configure authorization information and click **OK**.

After the authorization is complete, a list of authorized credentials will be displayed.

**Table 6-28** Authorization configuration

| Parameter         | Description  |
|-------------------|--|
| Environment       | Select the environment the API has been published in.  |
| Credentials       | Select the credentials you want to authorize.  |
| Access Parameters | Set access parameters for the selected credentials to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel     | Enabling Green Channel allows whitelisted clients to call the API without authentication.  |
| Whitelist         | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. Whitelisted clients can call the API without authentication.   |

| Parameter | Description  |
|-----------|--|
| Blacklist | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. Blacklisted clients are not allowed to call the API. |

## 6.4 Exposing a Data API

### 6.4.1 Connecting to a Data Source

#### Overview

Before creating a data API, connect to a data source to read data from. Access configuration varies depending on the data source type.

#### Prerequisites

- Before accessing a data source, ensure that the ROMA Connect instance can connect to the network your data source is in.
  - Same VPC: Let the instance directly access the data source.
  - Two VPCs in the same region: Connect the instance and the data source with a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.
- To enable cross-VPC private access, [configure the routes](#) between the instance and the subnet your data source is in.

#### Connecting to a Data Source

- [Connecting to a DWS Data Source](#)
- [Connecting to a DM Data Source](#)
- [Connecting to a Gauss100 Data Source](#)
- [Connecting to a HANA Data Source](#)
- [Connecting to a HIVE Data Source](#)
- [Connecting to a MongoDB Data Source](#)
- [Connecting to a MySQL Data Source](#)
- [Connecting to an Oracle Data Source](#)
- [Connecting to a PostgreSQL Data Source](#)
- [Connecting to a Redis Data Source](#)
- [Connecting to an SQL Server Data Source](#)

## 6.4.2 Creating a Data Backend

### Overview

Define data sources as backend services so that external systems using APIs can read or write the source data

### Constraints

- Each line of record stored in a data source should not exceed 2 KB, or custom backend exceptions will occur.
- If a numeric value returned by the data source is zero with more than six decimal places, the data backend displays the value in scientific notation. Do not set the precision of numeric data to more than six decimal places.

### Prerequisites

- Data sources are connected to ROMA Connect. For details, see [Connecting to a Data Source](#).
- If a backend service needs to use a signature key for authentication, [create a signature key](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, click **Create Backend**.
3. On the **Create Backend** page, set backend parameters and click **Create**. After the backend is created, the online IDE of the backend is automatically displayed. The backend type defaults to data backend.

**Table 6-29** Backend configuration

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a backend name. Using naming rules facilitates future search.  |
| Integration Application | Select an integration application for the backend. If none is available, click <b>Create Integration Application</b> on the right to create one. |
| Backend Request Method  | Options: <b>GET, POST, PUT, DELETE</b>   |
| Backend Request Path    | The path is case sensitive.<br>Example: <b>/getUserInfo/userId</b>   |

| Parameter                       | Description  |
|---------------------------------|--|
| Backend Security Authentication | Select the security authentication mode of the backend. <ul style="list-style-type: none"><li>● <b>Signature Key:</b> Authenticate backend requests using a signature key. The frontend API must be bound to the same signature key.</li><li>● <b>None:</b> No authentication is required for backend requests.</li></ul>  |
| Description                     | Enter a description of the backend.  |
| Advanced Settings               |  |
| Version                         | Enter the backend version to differentiate backend services.<br>Example: <b>V1.0</b>   |
| Input Parameters                | Define request parameters of the backend service in the <b>Input Parameters</b> area. Click <b>Add Input Parameter</b> and add: <ul style="list-style-type: none"><li>● <b>Name:</b> request parameter name</li><li>● <b>Parameter Location:</b> location of the request parameter in the backend request.<br/>Options: <b>Headers</b> or <b>Parameters</b></li><li>● <b>Default Value:</b> used only in the subsequent custom backend test procedure. This parameter does not take effect during custom backend deployment.</li><li>● <b>Mandatory:</b> whether a request parameter is mandatory in a backend request.</li><li>● <b>Description:</b> description of the request parameter</li></ul> |
| Returned Type                   | Select the response data format of the backend.<br>Options: <b>JSON, XML, STREAM</b>   |

4. Configure the data backend.
  - a. In the upper left corner, choose **New Data Backend > Add Data Source**.
  - b. On the **Add Data Source** page displayed, configure data source information and click **Add**.

**Table 6-30** Data source configuration

| Parameter          | Description   |
|--------------------|---|
| Select Data Source | Select a data source that you have created in <a href="#">Connecting to a Data Source</a> . |

| Parameter             | Description   |
|-----------------------|---|
| Select Statement Type | Select the type of a statement to be executed.<br>Options: <b>SQL</b> or <b>SP</b> (stored procedure)<br>If a Redis or MongoDB data source is used, select <b>SQL</b> . (NoSQL statements are actually used for these data sources).  |
| Advanced Settings     |   |
| Returned Object       | Enter the name of the object to be returned. The statement execution result is encapsulated in the object.  |
| Paging                | <p>Specify whether to return the statement execution result by page. This parameter is not available if multiple data sources have been selected for the data backend.</p> <p>Enabled: query parameters <b>pageNum</b> and <b>pageSize</b> can be added to the backend request to paginate the query results and specify the page number of the data records to be returned.</p> <ul style="list-style-type: none"><li>• <b>pageNum</b>: page number of the data records to be returned, starting from 1</li><li>• <b>pageSize</b>: number of data records on each page</li></ul> <p>The response structure varies depending on whether paging is enabled or disabled. For details, see <a href="#">Examples of paging results</a>.</p> <p><b>NOTE</b><br/>If there are more than 2000 records, try using the <b>offset</b> and <b>limit</b> parameters in the execution statement. If <b>Precompiling</b> is disabled, the following is an example:<br/>select * from table01 limit \${limit} offset \${offset}</p> <p>Keys of the <b>offset</b> and <b>limit</b> parameters can be transferred in the headers, parameters, or body of backend requests.</p> <p>Data sources with precompiling enabled need to convert <b>offset</b> and <b>limit</b> values by calling functions. For details, see <a href="#">Developing Custom Data Backends</a>.</p> |
| Precompiling          | Specify whether to precompile execution statements to prevent SQL injection attacks.  |

### Examples of paging results

**Returned Object** is set to **mydata**. After a statement is executed to query data from a data source, a total of five data records are returned.

- If **Paging** is not enabled, all five data records are returned as the response result to the user. The following is an example of the response result.

```
{  
  "mydata": [  
    {
```

```
"id": 1,  
  "name": "aaa"  
},  
{  
  "id": 2,  
  "name": "bbb"  
},  
{  
  "id": 3,  
  "name": "ccc"  
},  
{  
  "id": 4,  
  "name": "ddd"  
},  
{  
  "id": 5,  
  "name": "eee"  
}  
]  
}
```

- When **Paging** is enabled, if **pageNum** is set to **1** and **pageSize** is set to **2**, the five data records will be displayed on different pages based on **pageSize**, with two data records displayed on each page. Only the two data records on the first page will be returned to the user as the response result based on **pageNum**. In the response result, **total** indicates the total number of data records queried, that is, **5**.

Response example:

```
{  
  "mydata": {  
    "total": 5,  
    "data": [  
      {  
        "id": 1,  
        "name": "aaa"  
      },  
      {  
        "id": 2,  
        "name": "bbb"  
      }  
    ],  
    "pageSize": 2,  
    "pageNum": 1  
  }  
}
```

- c. After adding the data source, select the data source in the left pane of the online IDE, and then compile execution statements for the data source in the right pane.  
For details, see [Developing Custom Data Backends](#).  
For the Redis or MongoDB data source, data processing commands of Redis or MongoDB are used.
  - d. Click **Save** in the upper right corner of the page.
5. Test the backend functions.  
In the upper right corner of the page, click **Test**. In the **Test Parameters** area, add request parameters based on the backend definition and click **Test** to send the request.
    - In the **Execution Result** area, view the backend response.

- In the **Execution History** area, view the historical test records of the backend. Click a test record to import historical test parameters to the test parameter list on the left and perform the test again.
6. Deploy the backend.  
After the backend is tested, click **Deploy** in the upper right corner. In the dialog box displayed, click **Yes** to deploy the backend.

### 6.4.3 Publishing a Data API

After creating a data backend, publish it as an API in an environment. The generated API can be called by other users.

#### Prerequisites

- Each API must belong to an API group. Before publishing a function API, ensure that there are available API groups. Otherwise, [create an API group](#) first.
- If you need to use custom authentication, [create a frontend custom authorizer](#) first.

#### Constraints

If request parameters are added to the request path of a frontend API published by a custom backend, the API cannot be published on the publish page of the custom backend. You need to publish the API on the frontend API page. If just the content of the custom backend is modified, deploy the backend again and no publishing is required.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**.
3. In the backend list, click the name of a custom backend. The **Online IDE** page will be displayed.
4. Click **Publish** in the upper right corner of the page.
5. On the page displayed, configure API information and click **Publish** to create a frontend data API for the backend and publish the API in an environment.

**Table 6-31** Parameters for publishing an API

| Parameter | Description  |
|-----------|--|
| API Group | Select the API group to which the corresponding API belongs. If none is available, click <b>Create API Group</b> on the right to create one.<br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API. |

| Parameter                        | Description   |
|----------------------------------|---|
| Environment                      | Select the environment in which the API is to be published. You can also click <b>Create Environment</b> to create one.   |
| Frontend Security Authentication | Select the authentication mode of the API. The App authentication mode is recommended. <ul style="list-style-type: none"><li>● <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an integration application.</li><li>● <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK.</li><li>● <b>Custom</b>: The custom function API is used for authenticating API requests.</li><li>● <b>None</b>: Authentication is not required for API requests.</li></ul> |
| Custom Authorizer                | This parameter is mandatory only if <b>Frontend Security Authentication</b> is set to <b>Custom</b> .<br>Select a frontend custom authorizer you have created.  |
| Frontend Request Protocol        | Select the request protocol used by the frontend API. The value can be <b>HTTP</b> , <b>HTTPS</b> , or <b>HTTP&amp;HTTPS</b> . <b>HTTPS</b> is recommended for transmitting important or sensitive data.  |
| Timeout (ms)                     | Enter the timeout interval of a backend service request. The default value is <b>60000</b> .  |
| Retries                          | Number of retries after ROMA Connect fails to call the backend service. <ul style="list-style-type: none"><li>● If the value is <b>-1</b>, the retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>● If the value ranges from 0 to 10, this parameter is enabled, and retries are performed based on the configured value.</li></ul>   |
| Advanced Settings                |   |
| Frontend Request Method          | Select the request method of the frontend API. <b>ANY</b> indicates that the API can be accessed using any request method.  |
| Frontend Request Path            | Enter the request path of the frontend API, for example, <b>/getUserInfo/userId</b> .<br>The value of <b>Request Path</b> is case sensitive.  |



| Parameter | Description   |
|-----------|---|
| CORS      | This parameter specifies whether CORS is supported for the API.<br><br>For security purposes, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send <b>XMLHttpRequest</b> requests to a server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a> . |

## 6.4.4 Binding Domain Names

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are private or public.

- Private domain name: Service systems deployed on the cloud service platform use these names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform use these names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  - a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  - a. Apply for a public zone from Domain Registration.
  - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

## Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Groups**. Click the name of an API group to view details.
3. On the **Group Information** tab, click **Bind Independent Domain Name**.
4. In the displayed dialog box, enter the domain name and click **OK**.

**Table 6-32** Independent domain name configuration

| Parameter                      | Description   |
|--------------------------------|---|
| Domain Name                    | Enter the domain name to bind.  |
| Minimum TLS Version            | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected.  |
| HTTP-to-HTTPS Auto Redirection | Whether to support HTTP-to-HTTPS redirection.<br>Redirection takes effect only when the API request protocol is HTTPS or HTTP&HTTPS and an SSL certificate has been bound to the independent domain name.<br><b>NOTE</b><br>Redirection is only suitable for GET and HEAD requests. Redirecting other requests may cause data loss due to browser restrictions. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate to the independent domain name.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.
    - If a CA certificate has been configured for the SSL certificate, client authentication (HTTPS two-way authentication) is enabled by default.
    - If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

### 6.4.5 (Optional) Authorizing Credentials to Call APIs

Credentials need to be authorized to call APIs that use App authentication. When calling an API, a user is authenticated using the key and secret of a credential.

 **NOTE**

An API's integration application can directly call the API.

## Constraints

The security authentication mode of the API is set to **App**.

## Prerequisites

The API has been published in an environment. Otherwise, [publish the API](#) first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. Choose **More > Authorize Credentials** of an API.
3. On the page displayed, click **Select Credentials**.
4. Configure authorization information and click **OK**.

After the authorization is complete, a list of authorized credentials will be displayed.

**Table 6-33** Authorization configuration

| Parameter         | Description  |
|-------------------|--|
| Environment       | Select the environment the API has been published in.  |
| Credentials       | Select the credentials you want to authorize.  |
| Access Parameters | Set access parameters for the selected credentials to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel     | Enabling Green Channel allows whitelisted clients to call the API without authentication.  |
| Whitelist         | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. Whitelisted clients can call the API without authentication.   |
| Blacklist         | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. Blacklisted clients are not allowed to call the API.   |

## 6.5 Calling an API

## 6.5.1 Calling an Open API

After an API is published in an environment, it can be called by other users. API calling operations vary by the authentication mode used by the API.

### Constraints

- If you use the default subdomain name to access an API, you can access the API a maximum of 1000 times every day.
- If there are two APIs with the same group, request method, and request path, the API with exact matching is first called.

### Prerequisites

Before calling an API, ensure that the network of your service system can communicate with the API access domain name or address. This depends on where your service system is in relative to the ROMA Connect instance.

- The same VPC: Let the instance directly access the API.
- Two VPCs in the same region: Connect the instance and the service system with a peering connection. For details, see [VPC Peering Connection](#).
- Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
- Communication over the public network: Ensure that the instance must be bound with an EIP.

### Obtaining API Calling Information

Obtain the API calling information from the API provider before you call an API.

- Request information  
On the ROMA Connect instance console, choose **API Connect > APIs** and obtain the domain name, request method, and URL of an API. Click the API name to go to the details page. Click the **Frontend Configuration** tab on the **APIs** page, obtain the request parameters and request body of the API.

- Authentication information

Obtain the request authentication information according to the authentication mode used by an API.

| Authentication Mode                                | Authentication Info Needed   |
|--|--|
| App authentication (with a signature)              | Key and secret of the credential authorized by the API from the API provider, as well as the SDK, for authentication signatures. |
| App authentication (through simple authentication) | AppCode of the credential authorized by the API from the API provider.   |

| Authentication Mode                                  | Authentication Info Needed  |
|--|---|
| App authentication (with green channel whitelisting) | Key of the credential authorized by the API from the API provider.                    |
| App authentication (with app_secret)                 | Key and secret of the credential authorized by the API from the API provider.         |
| App authentication (with app_basic)                  | Key and secret of the credential authorized by the API from the API provider.         |
| App authentication (two-factor)                      | Authentication information required for App authentication and custom authentication. |
| IAM authentication (with a token)                    | Username and password of the cloud service platform.                                  |
| IAM authentication (with AK/SK)                      | AK/SK of the account on the cloud service platform and the SDK used for signatures.   |
| IAM authentication (two-factor)                      | Information required for IAM authentication and custom authentication.                |
| Custom authentication                                | Information to be carried in the request parameters from the API provider.            |
| None   | No information is required.   |

- Key and secret of a credential  
On the ROMA Connect instance console, choose **API Connect > Credentials**. Click the name of a credential authorized by the API. On the page displayed, obtain the key and secret of the credential.
- SDK used for authentication signatures  
On the ROMA Connect instance console, choose **API Connect > Credentials**. Download the SDK of the required language on the **SDKs** page.
- AppCodes  
On the ROMA Connect instance console, choose **API Connect > Credentials**. Click the name of a credential authorized by the API. On the page displayed, obtain an AppCode of the credential in the **AppCodes** area.

## Calling an API

1. Example API request

```
POST https://{Address}/{Path}?{Query}
{Header}

{
  {Body}
}
```

- *POST*: request method. Use the actual request method obtained in [Obtaining API request information](#).
- *{Address}*: request address. Use the actual request address obtained in [Obtaining API request information](#). You can also use an IP address to access an API.

| Scenario   | API Request Parameter Configuration  |
|--|--|
| Using a domain name to call an API                         | ROMA Connect allows APIs to be called using the subdomain name assigned to the API group or the domain name bound to the API group. No additional configuration is required.   |
| Calling an API in the DEFAULT group with an IP address     | Call an API in the DEFAULT group with an IP address. No additional configuration is required.  |
| Calling an API not in the DEFAULT group with an IP address | <ul style="list-style-type: none"> <li>• The <b>app_route</b> parameter described in <a href="#">Modifying Instance Configuration Parameters</a> has been set to <b>on</b> for a ROMA Connect instance, indicating that an API can be called by using an IP address.</li> <li>• ROMA Connect does not allow APIs in non-DEFAULT groups to be directly called using IP addresses. The header parameter <b>X-HW-ID</b> must be added to the request message, and the value must be the key of the credential authorized by the API.</li> </ul> |

- *{Path}*: request path. Use the actual URL obtained in [Obtaining API request information](#).
- (Optional) *{Query}*: query parameter in Parameter\_name=Parameter\_value format, for example, **limit=10**. Use & to separate multiple query parameters. For details, see the request parameters obtained in [Obtaining API request information](#).
- *{Header}*: request header parameter in Parameter\_name:Parameter\_value format, for example, **Content-Type:application/json**. For details, see the request parameters obtained in [Obtaining API request information](#).
- *{Body}*: request body in JSON format. For details, see the request body description obtained in [Obtaining API request information](#).

2. Add authentication information for the API request.

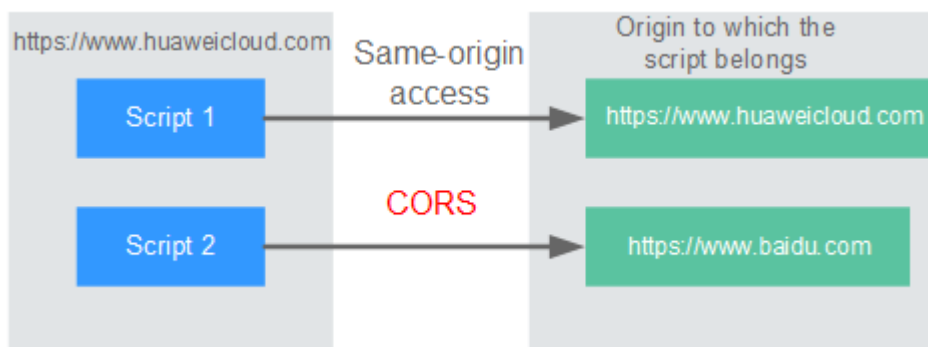
| API Authentication Mode                              | API Request Parameter Configuration   |
|--|---|
| App authentication (with a signature)                | Obtain the SDK to sign the API request. For details, see <a href="#">Developing API Calling Authentication (App)</a> .  |
| App authentication (through simple authentication)   | Add the header parameter <b>X-Apig-AppCode</b> to the API request. The parameter value is the AppCode obtained in <a href="#">Obtaining API authentication information</a> .  |
| App authentication (with green channel whitelisting) | Add the header parameter <b>X-HW-ID</b> to the API request. The parameter value is the key obtained in <a href="#">Obtaining API authentication information</a> .   |
| App authentication (with app_secret)                 | <ul style="list-style-type: none"><li>• The <b>app_secret</b> parameter has been set to <b>on</b> on the <a href="#">Configuration Parameters</a> tab page of a ROMA Connect instance, indicating that app_secret authentication is enabled.</li><li>• Add the header parameter <b>X-HW-ID</b> to the API request. The parameter value is the key obtained in <a href="#">Obtaining API authentication information</a>.</li><li>• Add the header parameter <b>X-HW-AppKey</b> to the API request. The parameter value is the secret obtained in <a href="#">Obtaining API authentication information</a>.</li></ul> |
| App authentication (with app_basic)                  | <ul style="list-style-type: none"><li>• The <b>app_basic</b> parameter described in <a href="#">Modifying Instance Configuration Parameters</a> has been set to <b>on</b> for a ROMA Connect instance, indicating that app_basic authentication is enabled.</li><li>• Add the header parameter <b>Authorization</b> to the API request. The value is "<b>Basic "+base64(appkey +":"+appsecret)</b>". <b>appkey</b> and <b>appsecret</b> are the key and secret obtained in <a href="#">Obtaining API authentication information</a>.</li></ul>  |
| App authentication (two-factor)                      | An API request carries authentication information of both App authentication and custom authentication.   |
| IAM authentication (with a token)                    | Obtain the authentication token from the cloud service platform, add the header parameter <b>X-Auth-Token</b> to the API request, and set the value to the authentication token. For details, see <a href="#">Token Authentication</a> .  |
| IAM authentication (with AK/SK)                      | Sign API requests using the obtained SDK. For details, see <a href="#">AK/SK Authentication</a> .   |

| API Authentication Mode         | API Request Parameter Configuration   |
|---------------------------------|---|
| IAM authentication (two-factor) | An API request carries authentication information of both IAM authentication and custom authentication.   |
| Custom authentication           | Based on the definition of custom authentication, the related authentication information is carried in the API request parameters for authentication. |
| None                            | No authentication is required, and the API can be directly called.  |

## 6.5.2 Configuring CORS for APIs

For security, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send cross-domain **XMLHttpRequest** requests.

Figure 6-3 Cross-domain access



CORS requests are either simple or non-simple.

- **Simple requests** meet both of the following conditions:
  - The request method is **HEAD**, **GET**, or **POST**.
  - The HTTP header can contain only the following fields: **Accept**, **Accept-Language**, **Content-Language**, **Last-Event-ID** and **Content-Type** (only three values are allowed: **application/x-www-form-urlencoded**, **multipart/form-data**, and **text/plain**).

In the header of a simple request, browsers automatically add the **Origin** field to specify the origin (including the protocol, domain, and port) of the request. After receiving such a request, the target server uses the origin to determine whether the request is safe and can be accepted. If the server sends a response containing the **Access-Control-Allow-Origin** field, the server accepts the request.

- **Non-simple requests** do not meet the preceding two conditions. Before sending a non-simple request, a browser first sends an HTTP request to the target server to determine whether the origin the web page is loaded



from is in the allowed origin list, and which HTTP request methods and header fields can be used. Once the HTTP request is successfully preflighted, the browser then sends a simple request to the server.

By default, ROMA Connect does not support CORS. To enable ROMA Connect to support CORS, enable CORS when [creating an API](#). For a non-simple CORS request, you also need to create an API that uses the **OPTIONS** method for preflight.

## Simple Requests

- **Scenario 1:** If CORS is enabled and the response from the backend does not contain a CORS header, ROMA Connect can handle requests from any domain, and returns the Access-Control-Allow-Origin CORS header. The following messages are used as examples:

a. **Request sent by a browser and containing the Origin header field:**

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

**Origin:** This field specifies the origin (<http://www.cors.com> in this example) of the request. It is mandatory. ROMA Connect and the backend service determine based on the origin whether the request is safe and can be accepted.

b. **Response sent by the backend:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
```

```
{"status": "200"}
```

c. **Response sent by ROMA Connect:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *
```

```
{"status": "200"}
```

**Access-Control-Allow-Origin:** This field is mandatory. The asterisk (\*) indicates that ROMA Connect accepts requests from any domain.

- **Scenario 2:** If CORS is enabled and the response from the backend contains a CORS header, the header will overwrite that added by ROMA Connect. The following messages are used as examples:

a. **Request sent by a browser and containing the Origin header field:**

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

**Origin:** This field specifies the origin (<http://www.cors.com> in this example) of the request. It is mandatory. ROMA Connect and the

backend service determine based on the origin whether the request is safe and can be accepted.

b. **Response sent by the backend:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
Access-Control-Allow-Origin: http://www.cors.com

{"status":"200"}
```

**Access-Control-Allow-Origin:** Indicates that the backend service accepts requests sent from **http://www.cors.com**.

c. **Response sent by ROMA Connect:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: http://www.cors.com

{"status":"200"}
```

The CORS header in the backend response overwrites that in ROMA Connect's response.

## Non-Simple Requests

For a non-simple request, you also need to create an API using the **OPTIONS** method. The request parameters of an API accessed using the **OPTIONS** method are:

- **Group:** The same group to which the API with CORS enabled belongs.
- **Security Authentication:** No authentication is required for requests received by the new API no matter which security authentication mode has been selected.
- **Protocol:** The same protocol used by the API with CORS enabled.
- **Request Path:** The same or matching request path used by the API with CORS enabled.
- **Method:** Set to **OPTIONS**.
- **CORS:** Enabled.
- **Backend service:** Returns 200 OK as the response.

The following are example requests and responses sent to or from a mock backend.

1. **Request sent from a browser to an API that is accessed using the OPTIONS method:**

```
OPTIONS /HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost
Accept: */*
Origin: http://www.cors.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Sdk-Date
```

- **Origin:** This field is mandatory and used to specify the origin from which the request has been sent.

- **Access-Control-Request-Method:** This field is mandatory and used to specify the HTTP methods to be used by the subsequent simple requests.
  - **Access-Control-Request-Headers:** This field is mandatory and used to specify the additional header fields in the subsequent simple requests.
2. **Response sent by the backend:**  
None
3. **Response sent by ROMA Connect:**
- ```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 02:38:48 GMT
Content-Type: application/json
Content-Length: 1036
Server: roma
X-Request-Id: c9b8926888c356d6a9581c5c10bb4d11
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Stage,X-Sdk-Date,X-Sdk-Nonce,X-Proxy-Signed-Headers,X-Sdk-Content-Sha256,X-Forwarded-For,Authorization,Content-Type,Accept,Accept-Ranges,Cache-Control,Range
Access-Control-Expose-Headers: X-Request-Id,X-Apig-Latency,X-Apig-Upstream-Latency,X-Apig-RateLimit-Api,X-Apig-RateLimit-User,X-Apig-RateLimit-App,X-Apig-RateLimit-Ip,X-Apig-RateLimit-Api-Allenv
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH
Access-Control-Max-Age: 172800
```
- **Access-Control-Allow-Origin:** This field is mandatory. The asterisk (\*) indicates that ROMA Connect accepts requests from any domain.
  - **Access-Control-Allow-Headers:** This field is required if it is contained in the request. It specifies the header information field supported by ROMA Connect.
  - **Access-Control-Allow-Methods:** This field is mandatory and used to specify the HTTP request method supported by ROMA Connect.
  - **Access-Control-Max-Age:** This field is mandatory and used to specify the period (in seconds) during which the preflight result remains valid. No more preflight requests are needed within the period.
4. **Request sent by a browser and containing the Origin header field:**
- ```
PUT /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```
- Origin:** This field is mandatory and used to specify the origin from which the request has been sent.
5. **Response sent by the backend:**
- ```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma

{"status":"200"}
```
6. **Response sent by APIC:**
- ```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *

{"status":"200"}
```

## 6.5.3 Viewing API Calling Statistics

View API calling statistics on the ROMA Connect console.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Monitoring & Analysis**. On the **API Monitoring** tab, view the API calling statistics, which include:
  - Real-time statistics on the total number of APIs, API groups, and request throttling policies
  - Number of requests, number of errors, data traffic, and latency of an APIYou can also specify an integration application, API, and time range to view data.
  - Data in the last hour is updated every 2 minutes.
  - Data in the last 6 hours is updated every 2 hours.
  - Data in the last day is updated every 2 hours.
  - Data in the last week and last month is updated every day.

## 6.5.4 Viewing API Call Logs

View API call logs on the ROMA Connect console.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Monitoring & Analysis**.
3. Enable log analysis.
  - a. On the **Log Analysis** tab page, click **Configure Log Collection**.
  - b. In the dialog box displayed, configure log collection information and click **OK**.

**Table 6-34** Parameters for configuring access logs

| Parameter    | Description  |
|--------------|--|
| Collect Logs | Determine whether to enable logging. API call logs can be viewed only after you enable this parameter. |

| Parameter  | Description  |
|------------|--|
| Log Group  | Select the log group to which the log stream belongs.<br>If no log group is available, click <b>View Log Group</b> to switch to the LTS console and create a log group. For details, see <a href="#">Creating a Log Group</a> .    |
| Log Stream | Select the log stream for storing the API call logs.<br>If no log stream is available, click <b>View Log Stream</b> to switch to the LTS console and create a log stream. For details, see <a href="#">Creating a Log Stream</a> . |

- After the log analysis function is enabled, view the call logs of all open APIs in real time on the console. For details about the log fields, see [Log Field Description](#).
  - In the upper right corner of the page, select a time segment.
  - Click **View Log Details** to go to the LTS console to view log details and download logs locally.

Custom fields in log details:

- \_resource\_id**: ROMA Connect instance ID
- \_service\_type**: source service

## Log Field Description

| No. | Field           | Description   |
|-----|-----------------|---|
| 1   | remote_addr     | Client IP address   |
| 2   | request_id      | Request ID  |
| 3   | api_id          | API ID  |
| 4   | user_id         | Project ID provided by a requester for IAM authentication.      |
| 5   | app_id          | Application ID provided by a requester using App authentication |
| 6   | time_local      | Request time  |
| 7   | request_time    | Request latency, in seconds                                     |
| 8   | request_method  | HTTP request method   |
| 9   | host            | Request domain name   |
| 10  | router_uri      | Request URI   |
| 11  | server_protocol | Request protocol  |
| 12  | status          | Response code   |

| No. | Field                      | Description  |
|-----|----------------------------|--|
| 13  | bytes_sent                 | Response size (including the status line, response header, and response body), in bytes  |
| 14  | request_length             | Request length (including the start line, request header, and request body), in bytes  |
| 15  | http_user_agent            | User agent ID  |
| 16  | http_x_forwarded_for       | X-Forwarded-For header field   |
| 17  | upstream_addr              | Backend address  |
| 18  | upstream_uri               | Backend URI  |
| 19  | upstream_status            | Backend response code  |
| 20  | upstream_connect_time      | Time taken for establishing a connection with the backend  |
| 21  | upstream_header_time       | Duration from the beginning of the establishment of a connection to receiving the first byte from the backend, in seconds  |
| 22  | upstream_response_time     | Duration from the beginning of the establishment of a connection to receiving the last byte from the backend, in seconds   |
| 23  | region_id                  | AZ ID  |
| 24  | all_upstream_response_time | Duration from the beginning of the establishment of a connection to receiving the last byte from the backend, in seconds. When a retry occurs, the value is the total time used. |
| 25  | errorType                  | Request error type. Options: <ul style="list-style-type: none"> <li>• <b>0</b>: non-throttling error</li> <li>• <b>1</b>: throttling error</li> </ul>                            |
| 26  | auth_type                  | API authentication mode  |
| 27  | access_model1              | Authentication mode 1  |
| 28  | access_model2              | Authentication mode 2. When two-factor authentication is enabled, the custom authorizer ID is used.  |
| 29  | inner_time                 | Internal processing duration of APIC, in seconds   |
| 30  | proxy_protocol_vni         | VPC endpoint virtual network ID  |

| No. | Field                  | Description   |
|-----|------------------------|---|
| 31  | proxy_protocol_vpce_id | VPC endpoint ID   |
| 32  | proxy_protocol_addr    | Client IP address   |
| 33  | body_bytes_sent        | Size of the API request body, in bytes  |
| 34  | api_name               | API name  |
| 35  | app_name               | Name of the application used by the requester when App authentication is used   |
| 36  | provider_app_id        | ID of the application to which the API belongs  |
| 37  | provider_app_name      | Name of the application to which the API belongs  |
| 38  | custom_data_log1       | Custom log field 1  |
| 39  | custom_data_log2       | Custom log field 2  |
| 40  | custom_data_log3       | Custom log field 3  |
| 41  | custom_data_log4       | Custom log field 4  |
| 42  | custom_data_log5       | Custom log field 5  |
| 43  | custom_data_log6       | Custom log field 6  |
| 44  | custom_data_log7       | Custom log field 7  |
| 45  | custom_data_log8       | Custom log field 8  |
| 46  | custom_data_log9       | Custom log field 9  |
| 47  | custom_data_log10      | Custom log field 10   |
| 48  | response_source        | Request response source. Options: <ul style="list-style-type: none"><li>• <b>local</b>: APIG</li><li>• <b>remote</b>: backend service</li></ul> |

## 6.5.5 Appendix: API Error Codes

The following table lists the error codes returned when a user fails to call an API.

**Table 6-35** Error codes

| HTTP Status Code | Error Code | Description  | Solution  |
|------------------|------------|--|---|
| 404              | APIC.0101  | The API does not exist or has not been published in the environment. | <ul style="list-style-type: none"><li>• Check whether the domain name, method, and path are consistent with those of the registered API.</li><li>• Check whether the API is published. If the API is published in a non-production environment, check whether the X-Stage header in the request is the name of the environment.</li></ul> |
| 500              | APIC.0103  | The backend does not exist.  | Contact technical support.  |
| 500              | APIC.0104  | The plug-ins do not exist.   | Contact technical support.  |
| 500              | APIC.0105  | The backend configurations do not exist.                             | Contact technical support.  |
| 400              | APIC.0106  | Orchestration error.   | Check whether the frontend and backend parameters of the API are correct.   |
| 400              | APIC.0201  | Bad request.   | Set valid request parameters.   |
| 413              | APIC.0201  | Request entity too large.  | Reduce the size of the request body to less than 12 MB.   |
| 414              | APIC.0201  | Request URI too large.   | Reduce the size of the URI to less than 32 KB.  |
| 494              | APIC.0201  | Request headers too large.   | Reduce the size of request headers to ensure that the length of a single request header is less than 32 KB or the total length of all request headers is less than 128 KB.  |
| 502              | APIC.0202  | Backend unavailable.   | Check whether the backend address configured for the API is accessible.   |
| 504              | APIC.0203  | Backend timeout.   | Increase the timeout duration of the backend service or shorten the processing time.  |



| HTTP Status Code | Error Code | Description   | Solution  |
|------------------|------------|---|---|
| 401              | APIC.0301  | Incorrect IAM authentication information.           | Check whether the token is correct.   |
| 403              | APIC.0302  | The IAM user is not authorized to access the API.   | Check whether the user is restricted by a blacklist or whitelist.   |
| 401              | APIC.0303  | Incorrect App authentication information.           | <ul style="list-style-type: none"> <li>• Check whether the request method, path, query parameters, and request body are consistent with those used for signature.</li> <li>• Check whether the client time is correct.</li> </ul> |
| 403              | APIC.0304  | The app is not authorized to access the API.        | Check whether the app has been authorized to access the API.  |
| 401              | APIC.0305  | Incorrect authentication information.               | Check whether the authentication information is correct.  |
| 403              | APIC.0306  | API access denied.                                  | Check whether you have been authorized to access the API.   |
| 401              | APIC.0307  | The token must be updated.                          | Update the token.   |
| 429              | APIC.0308  | The throttling threshold has been reached.          | Wait until the request throttling period ends and access the API again, or change the request throttling threshold.   |
| 403              | APIC.0401  | Unknown client IP address.                          | Contact technical support.  |
| 403              | APIC.0402  | The IP address is not authorized to access the API. | Check whether the IP address is restricted by a blacklist or whitelist.   |
| 503              | APIC.0404  | Access to the backend IP address has been denied.   | Use an available IP address to access the backend service.  |
| 403              | APIC.0405  | The app is not accessed from a trusted IP address.  | Check whether the IP address is restricted by the client access control policy.   |
| 500              | APIC.0601  | Internal server error.                              | Contact technical support.  |

| HTTP Status Code | Error Code | Description                                | Solution  |
|------------------|------------|--|---|
| 400              | APIC.0602  | Bad request.                               | Check whether the request is valid.   |
| 500              | APIC.0605  | Backend domain name resolution failed.     | Check whether the domain name is correct and has been bound to a correct backend address.                                 |
| 500              | APIC.0606  | Failed to load the API configurations.     | Contact technical support.  |
| 400              | APIC.0607  | The following protocol is supported: {xxx} | Use HTTP or HTTPS to access the API.  |
| 500              | APIC.0608  | Failed to obtain the admin token.          | Contact technical support.  |
| 500              | APIC.0609  | The VPC backend does not exist.            | Contact technical support.  |
| 502              | APIC.0610  | No backend available.                      | Check whether all backends are available.   |
| 500              | APIC.0611  | The backend port does not exist.           | Contact technical support.  |
| 500              | APIC.0612  | An API cannot call itself.                 | Modify the backend configurations, and ensure that the number of layers the API is recursively called does not exceed 10. |
| 500              | APIC.0705  | Backend signature calculation failed.      | Contact technical support.  |

## 6.5.6 Response Headers

The following table describes the response headers that APIC adds to the response returned when an API is called.

**X-Apig-Mode: debug** indicates APIC debugging information.

| Response Header | Description | Remarks                          |
|-----------------|-------------|----------------------------------|
| X-Request-Id    | Request ID. | Returned for all valid requests. |

| Response Header             | Description  | Remarks  |
|-----------------------------|--|--|
| X-Apig-Latency              | Duration from the time when APIC receives a request to the time when the backend returns a message header.             | Returned only when the request header contains <b>X-Apig-Mode: debug</b> .   |
| X-Apig-Upstream-Latency     | Duration from the time when APIC sends a request to the backend to the time when the backend returns a message header. | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and the backend type is not Mock.   |
| X-Apig-RateLimit-api        | API request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                                 | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called.                  |
| X-Apig-RateLimit-user       | User request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                                | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by a user.        |
| X-Apig-RateLimit-app        | Credential request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                          | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by a credential.  |
| X-Apig-RateLimit-ip         | IP address request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                          | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by an IP address. |
| X-Apig-RateLimit-api-allenv | Default API request limit information.<br>Example:<br><b>remain:199,limit:200,time:1 second.</b>                       | Returned only when the request header contains <b>X-Apig-Mode: debug</b> .   |
| X-Apig-count                | Total number of times that a request is forwarded by APIC.   | Returned for all valid requests forwarded by APIC. If the value of <b>X-Apig-count</b> is greater than 10, error APIC.0612 is reported.                                  |

## 6.6 Managing APIs

### 6.6.1 Taking an API Offline

If a published API is not needed to provide services, take it offline from its published environment.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. Choose **More > Take Offline** of an API.
3. On the page displayed, select the environment from which the API is to be taken offline and click **OK**.

#### NOTICE

Taking an API offline will make the API inaccessible in the environment. Before performing this operation, ensure that you have notified the users who have used this API.

### 6.6.2 Importing and Exporting APIs

ROMA Connect allows you to import and export APIs using JSON files. When APIs are imported, file content must comply with the Swagger 2.0 or specifications.

#### Prerequisites

- Supplement the [extended Swagger definition of APIs](#) in the API file to import.
- Ensure that the quotas of APIs and API groups meet the requirements before importing APIs.
- If you choose **New Group** when importing an API, the value of the **info.title** field in the API definition file is used as the API group name. Before importing an API, do not change the value of the **info.title** field.

#### Importing APIs

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. In the upper right corner of the page, click **Import APIs**.  
To import APIs, you can also choose **API Connect > API Groups**. In the upper right corner, click **Create API Group** and select **Import API Design File**.
3. Select a local Swagger file in YAML or JSON format.

## 4. Configure API import information.

**Table 6-36** Importing APIs

| Parameter                     | Description  |
|-------------------------------|--|
| Import To                     | Select the method for importing APIs. <ul style="list-style-type: none"><li>• <b>New group</b>: Import APIs to a new API group. If you select this option, the system automatically creates an API group and imports the APIs to this group.</li><li>• <b>Existing group</b>: Select an existing API group and add the imported APIs to this group.</li></ul>  |
| Scope                         | Mandatory for <b>Import To</b> set to <b>New group</b> .<br>Specify who can view the API group. <ul style="list-style-type: none"><li>• <b>Integration application</b>: An API group belongs to a specific integration application. Only users who have permissions for this application can view and perform operations on the API group.</li><li>• <b>All</b>: All users in the current instance can view and perform operations on the API group.</li></ul>                                   |
| Integration Application       | Mandatory for <b>Scope</b> set to <b>Integration application</b> .<br>Select the API group's integration application.  |
| Basic Definition Overwrite    | Available for <b>Import To</b> set to <b>Existing group</b> .<br>This parameter specifies whether to overwrite the existing APIs when the imported APIs conflict with those in the existing API group.   |
| Extended Definition Overwrite | Specifies whether to overwrite the existing extended information, such as custom authentication, request throttling policy, and access control policy, when there is a conflict between the imported API and ROMA Connect. <ul style="list-style-type: none"><li>• <b>Enable</b>: The existing extended information is overwritten with that defined in the imported file.</li><li>• <b>Disable</b>: The existing extended information is used, rather than that in the imported file.</li></ul> |
| Parameter Import              | Check the content of the imported file. <ul style="list-style-type: none"><li>• <b>Check</b> for valid format.</li><li>• <b>Format</b> the file.</li><li>• <b>Download</b> the file to the local host.</li><li>• <b>Enable Mock</b> to use a mock backend service for importing APIs.</li></ul>  |

5. Click **Import Now**.

To unify request and backend information of the imported APIs before the import, click **Configure Global Settings** and configure the information as prompted. Click **Submit** to import the APIs.

6. In the dialog box that is displayed, choose whether to publish the APIs now. If you select **Now**, select the environment you want to publish the APIs in.
7. Click **OK**.

## Exporting APIs

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > APIs**. In the upper right corner of the page, click **Export APIs**.
3. Configure API export information.

**Table 6-37** Parameters for exporting APIs

| Parameter      | Description   |
|----------------|---|
| API Group      | Select the API group to export the APIs from.   |
| Environment    | Select the environment to export the APIs from.   |
| APIs           | Select the APIs to be exported. If this parameter is not specified, all APIs in the API group in the selected environment will be exported by default.  |
| API Definition | Select the scope of the API definition to be exported. <ul style="list-style-type: none"><li>● <b>Basic</b>: Only the API frontend request information is exported. The API frontend request information includes both basic and extended Swagger fields.</li><li>● <b>Full</b>: Both API frontend request information and backend service information.</li><li>● <b>Extended</b>: The API frontend request information, backend service information, as well as the request throttling policy and access control policy associated with the API.</li></ul> |
| Format         | Select the format of the exported API file.<br>Options: <b>YAML</b> or <b>JSON</b>  |
| Version        | Enter the version of the API file to be exported. If no version is specified, the version will be set to the current time.  |

4. Click **Export** to export the API file to a local directory. The content of the exported file is displayed in the right pane.

 NOTE

- If no independent domain name is bound to the API group the exported API belongs to, the subdomain name of the API group will be exported.
- If multiple independent domain names are bound to the API group the exported API belongs to, only one independent domain name will be exported randomly.

### 6.6.3 Adding an SSL Certificate

If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. An SSL certificate is used for data encryption and identity authentication. It supports one-way and two-way authentication.

- One-way authentication: When connecting to a server, a client verifies whether the server is correct.
- Two-way authentication: When connecting to a server, a client verifies the server and the server also verifies the client.

#### Constraints

- Only SSL certificates in PEM format can be added.
- The added SSL certificates support only the RSA, ECDSA, and DSA encryption algorithms.
- Certificate chains are not supported.

#### Creating an SSL Certificate

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **SSL Certificates** tab, click **Create SSL Certificate**.
3. In the displayed dialog box, configure the SSL certificate information.

**Table 6-38** SSL certificate configuration

| Parameter | Description   |
|-----------|---|
| Name      | Enter an SSL certificate name. Using naming rules facilitates future search.  |
| Scope     | Specify the scope to view the SSL certificate. <ul style="list-style-type: none"><li>• <b>Current instance:</b> The SSL certificate can be viewed only in the current instance.</li><li>• <b>All:</b> The SSL certificate can be viewed in all instances under the current account.</li></ul> |

| Parameter | Description  |
|-----------|--|
| Content   | <p>Enter the SSL certificate content in PEM format.</p> <p>Open the PEM certificate file in the certificate to upload in text, and copy the certificate content to <b>content</b>.</p> <p>If the certificate is not in PEM encoding format, convert the format by referring to <a href="#">Converting the Certificate Format to PEM</a>.</p>   |
| Key       | <p>Enter the SSL certificate key in PEM format.</p> <p>Open the KEY/PEM private key file in the certificate to be uploaded in text, and copy the private key to <b>Key</b>.</p>  |
| CA        | <p>When two-way authentication is used, a CA certificate is required to verify the client certificate. After configuring a CA certificate, <a href="#">bind an SSL certificate</a> to the independent domain name to enable two-way authentication.</p> <p>Open (in text mode) the CA certificate file (.pem format) of the certificate to be uploaded and copy the content to <b>CA</b>.</p> <p>If the certificate is not PEM-coded, convert the format by referring to <a href="#">Converting the Certificate Format to PEM</a>.</p> |

4. Click **OK**. The SSL certificate is added.

## Converting the Certificate Format to PEM

| Format  | Converting with <a href="#">OpenSSL</a>  |
|---------|--|
| CER/CRT | Rename the certificate file <b>cert.crt</b> to <b>cert.pem</b> directly.   |
| PFX     | <ul style="list-style-type: none"><li>• Obtain a private key. For example, run the following command to convert <b>cert.pfx</b> into <b>key.pem</b>:<br/>openssl pkcs12 -in cert.pfx -nocerts -out key.pem</li><li>• Obtain a certificate. For example, run the following command to convert <b>cert.pfx</b> into <b>cert.pem</b>:<br/>openssl pkcs12 -in cert.pfx -nokeys -out cert.pem</li></ul> |
| P7B     | <ol style="list-style-type: none"><li>1. Convert a certificate. For example, run the following command to convert <b>cert.p7b</b> into <b>cert.cer</b>:<br/>openssl pkcs7 -print_certs -in cert.p7b -out cert.cer</li><li>2. Rename the certificate file <b>cert.cer</b> to <b>cert.pem</b>.</li></ol>   |



| Format | Converting with <a href="#">OpenSSL</a>   |
|--------|---|
| DER    | <ul style="list-style-type: none"><li>Obtain a private key. For example, run the following command to convert <b>privatekey.der</b> into <b>privatekey.pem</b>:<br/><b>openssl rsa -inform DER -outform PEM -in privatekey.der -out privatekey.pem</b></li><li>Obtain a certificate. For example, run the following command to convert <b>cert.der</b> into <b>cert.pem</b>:<br/><b>openssl x509 -inform der -in cert.der -out cert.pem</b></li></ul> |

## 6.6.4 Adding a Credential for Simple Authentication

If simple authentication is enabled for an API, AppCodes configured in a credential can be used for authentication. In this case, keys and secrets are not required.

### Constraints

Only APIs using App authentication support simple authentication.

### Procedure

- Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
- In the navigation pane on the left, choose **API Connect** > **Credentials**. Click the name of a credential authorized by the API.
- In the **AppCodes** area, click **Add AppCode**.
- In the dialog box displayed, configure AppCode information.

**Table 6-39** AppCode configuration

| Parameter    | Description  |
|--------------|--|
| AppCode Type | Select the method for generating an AppCode. <ul style="list-style-type: none"><li><b>Automatically generated</b>: An AppCode is generated by the system.</li><li><b>Custom</b>: Specify an AppCode.</li></ul> |
| AppCode      | Mandatory for <b>AppCode Type</b> set to <b>Custom</b> .<br>Enter an AppCode value.  |

- Click **OK**.

## 6.6.5 Appendix: Extended Swagger Definition of APIs

On the basis of the basic Swagger definition, ROMA Connect defines the extended definition of APIs, such as the authentication mode and backend service definition. This section describes the extended definitions of APIs.

## 1: x-apigateway-auth-type

**Meaning:** Swagger-based apiKey authentication format, which defines an authentication mode provided by ROMA Connect.

**Scope of effect:** [Security Scheme Object](#)

### Example

```
securityDefinitions:
  customize-name-iam:
    type: "apiKey"
    name: "unused"
    in: "header"
    x-apigateway-auth-type: "IAM"
  customize-name-app:
    type: "apiKey"
    name: "Authorization"
    in: "header"
    x-apigateway-auth-type: "AppSigv1"
  customize-name-iam-none:
    type: "apiKey"
    name: "unused"
    in: "header"
    x-apigateway-auth-type: "IAM_NONE"
```

**Table 6-40** Parameter description

| Parameter              | Man<br>dator<br>y | Type   | Description  |
|------------------------|-------------------|--------|--|
| type                   | Yes               | String | Authentication type. Only <b>apiKey</b> is supported.  |
| name                   | Yes               | String | Name of the parameter for authentication. <ul style="list-style-type: none"><li>When <b>x-apigateway-auth-type</b> is set to <b>AppSigv1</b>, set <b>name</b> to <b>Authorization</b>.</li><li>When <b>x-apigateway-auth-type</b> is set to <b>IAM</b> or <b>IAM_NONE</b>, set <b>name</b> to <b>unused</b>.</li></ul> |
| in                     | Yes               | String | Location of the parameter. Only <b>header</b> is supported.  |
| description            | No                | String | Description of the parameter.  |
| x-apigateway-auth-type | Yes               | String | APIC authentication mode. <b>AppSigv1</b> , <b>IAM</b> , and <b>IAM_NONE</b> are supported.  |

## 2: x-apigateway-request-type

**Meaning:** API request type, which can be **public** or **private**.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-request-type: 'public'
```

**Table 6-41** Parameter description

| Parameter                 | Mandatory | Type   | Description   |
|---------------------------|-----------|--------|---|
| x-apigateway-request-type | Yes       | String | API visibility. The options include <b>public</b> and <b>private</b> . <ul style="list-style-type: none"><li>• <b>public</b>: The API can be made available for sale.</li><li>• <b>private</b>: The API will not be available for sale.</li></ul> |

### 3: x-apigateway-match-mode

**Meaning:** Request URL matching mode, which can be **NORMAL** or **SWA**.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-match-mode: 'SWA'
```

**Table 6-42** Parameter description

| Parameter               | Mandatory | Type   | Description   |
|-------------------------|-----------|--------|---|
| x-apigateway-match-mode | Yes       | String | Matching mode of the API request path. The value can be <b>SWA</b> or <b>NORMAL</b> . <ul style="list-style-type: none"><li>• <b>SWA</b>: prefix match. For example, if the request path is set to <b>/test/AA</b> and the matching mode is set to <b>SWA</b>, the API can be accessed using <b>/test/AA/BB</b> or <b>/test/AA/CC</b> but cannot be accessed using <b>/test/AACC</b>.</li><li>• <b>NORMAL</b>: exact match.</li></ul> |

## 4: x-apigateway-cors

**Meaning:** Specifies whether APIs defined by ROMA Connect support CORS.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-cors: true
```

**Table 6-43** Parameter description

| Parameter         | Mandatory | Type    | Description                |
|-------------------|-----------|---------|----------------------------|
| x-apigateway-cors | No        | Boolean | Whether CORS is supported. |

For the API request for enabling CORS, the headers listed in the following table will be added to the response.

| Parameter Name               | Parameter Value  | Description  |
|------------------------------|--|--|
| Access-Control-Max-Age       | 172800   | Maximum time the response of a preflight request can be cached.                                    |
| Access-Control-Allow-Origin  | *  | Domain that can be accessed. The asterisk (*) indicates that requests from any domain are allowed. |
| Access-Control-Allow-Headers | X-Sdk-Date, X-Sdk-Nonce, X-Proxy-Signed-Headers, X-Sdk-Content-Sha256, X-Forwarded-For, Authorization, Content-Type, Accept, Accept-Ranges, Cache-Control, Range | Header information fields that can be used in API requests.  |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH   | HTTP methods allowed by the API request.   |

## 5: x-apigateway-any-method

**Meaning:** API request method used by default if no HTTP request method is specified.

**Scope of effect:** [Path Item Object](#)

### Example

```
paths:
  '/path':
    get:
      produces:
        - application/json
      responses:
        "200":
          description: "get response"
    x-apigateway-any-method:
      produces:
        - application/json
      responses:
        "200":
          description: "any response"
```

## 6: x-apigateway-backend

**Meaning:** API backend definition.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "backend endpoint type"
```

**Table 6-44** Parameter description

| Parameter        | Mandatory | Type  | Description  |
|------------------|-----------|---|--|
| type             | Yes       | String  | Backend service type. <b>HTTP</b> , <b>HTTP-VPC</b> , and <b>MOCK</b> are supported. |
| parameters       | No        | <a href="#">x-apigateway-backend.parameters</a>       | Backend parameters.  |
| httpEndpoints    | No        | <a href="#">x-apigateway-backend.httpEndpoints</a>    | HTTP backend service definition.   |
| httpVpcEndpoints | No        | <a href="#">x-apigateway-backend.httpVpcEndpoints</a> | HTTP-VPC backend service definition.   |

| Parameter         | Mandatory | Type                                   | Description                          |
|-------------------|-----------|--|--------------------------------------|
| functionEndpoints | No        | x-apigateway-backend.functionEndpoints | Function backend service definition. |
| mockEndpoints     | No        | x-apigateway-backend.mockEndpoints     | Mock backend service definition.     |

## 6.1: x-apigateway-backend.parameters

**Meaning:** API backend definition.

**Scope of effect:** x-apigateway-backend

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
        - name: "userId"
          description: "user name"
          type: "string"
          in: "path"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "HTTP"
        parameters:
          - name: "userId"
            value: "userId"
            in: "query"
            origin: "REQUEST"
            description: "user name"
          - name: "X-Invoke-User"
            value: "apigateway"
            in: "header"
            origin: "CONSTANT"
            description: "invoke user"
```

**Table 6-45** Parameter description

| Parameter   | Man<br>dator<br>y | Type   | Description   |
|-------------|-------------------|--------|---|
| name        | Yes               | String | Parameter name, which consists of a maximum of 32 bytes, starting with a letter. Only letters, digits, periods (.), hyphens (-), and underscores (_) are allowed.<br><br>The names of header parameters are not case-sensitive. |
| value       | Yes               | String | Parameter value, which is a parameter name if the parameter comes from a request.   |
| in          | Yes               | String | Parameter location, which can be <b>header</b> , <b>query</b> , or <b>path</b> .  |
| origin      | Yes               | String | Parameter mapping source. <b>REQUEST</b> and <b>CONSTANT</b> are supported.   |
| description | No                | String | Parameter description.  |

## 6.2: x-apigateway-backend.httpEndpoints

**Meaning:** HTTP backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "HTTP"
        httpEndpoints:
          address: "www.example.com"
          scheme: "http"
          method: "GET"
          path: "/users"
          retryCount: "3"
          timeout: 30000
```

**Table 6-46** Parameter description

| Parameter  | Mandatory | Type   | Description   |
|------------|-----------|--------|---|
| address    | Yes       | Array  | Backend service address. The format is <i>&lt;Domain name or IP address&gt;:[Port number]</i>   |
| scheme     | Yes       | String | Backend request protocol. <b>HTTP</b> and <b>HTTPS</b> are supported.   |
| method     | Yes       | String | Backend request method. <b>GET</b> , <b>POST</b> , <b>PUT</b> , <b>DELETE</b> , <b>HEAD</b> , <b>OPTIONS</b> , <b>PATCH</b> , and <b>ANY</b> are supported. |
| path       | Yes       | String | Backend request path, which can contain variables.  |
| retryCount | No        | String | Number of retry times upon a backend failure.   |
| timeout    | No        | Number | Backend request timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> .  |

### 6.3: x-apigateway-backend.httpVpcEndpoints

**Meaning:** HTTP-VPC backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

#### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "HTTP-VPC"
      httpVpcEndpoints:
        name: "vpc-test-1"
        scheme: "http"
        method: "GET"
        path: "/users"
        timeout: 30000
```



**Table 6-47** Parameter description

| Parameter | Mandatory | Type   | Description  |
|-----------|-----------|--------|--|
| name      | Yes       | Array  | VPC channel name.  |
| scheme    | Yes       | String | Backend request protocol. <b>HTTP</b> and <b>HTTPS</b> are supported.  |
| method    | Yes       | String | Backend request method. <b>GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH,</b> and <b>ANY</b> are supported.         |
| path      | Yes       | String | Backend request path, which can contain variables.   |
| timeout   | No        | Number | Backend request timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> . |

## 6.4: x-apigateway-backend.functionEndpoints

**Meaning:** FUNCTION backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "FUNCTION"
        functionEndpoints:
          version: "v1"
          function-urn: ""
          invocation-type: "synchronous"
          timeout: 30000
```

**Table 6-48** Parameter description

| Parameter       | Mandatory | Type   | Description   |
|-----------------|-----------|--------|---|
| function-urn    | Yes       | String | Function URN.   |
| version         | Yes       | String | Function version.   |
| invocation-type | Yes       | String | Function invocation type. <b>async</b> or <b>sync</b> are supported.  |
| timeout         | No        | Number | Function timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> . |

## 6.5: x-apigateway-backend.mockEndpoints

**Meaning:** Mock backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "MOCK"
        mockEndpoints:
          result-content: "mocked"
```

**Table 6-49** Parameter description

| Parameter      | Mandatory | Type   | Description    |
|----------------|-----------|--------|----------------|
| result-content | Yes       | String | Mock response. |

## 7: x-apigateway-backend-policies

**Meaning:** API backend policy.

**Scope of effect: Operation Object**

**Example**

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-apigateway-request-type: "public"
      x-apigateway-backend:
        type: "backend endpoint type"
      x-apigateway-backend-policies:
        - type: "backend endpoint type"
          name: "backend policy name"
      conditions:
        - type: "equal/enum/pattern",
          value: "string",
          origin: "source/request_parameter",
          parameter_name: "string"
```

**Table 6-50** Parameter description

| Parameter         | Man<br>dator<br>y | Type  | Description  |
|-------------------|-------------------|---|--|
| type              | Yes               | String  | Backend service type. <b>HTTP</b> , <b>HTTP-VPC</b> , and <b>MOCK</b> are supported. |
| name              | Yes               | String  | Name   |
| parameters        | No                | <b>x-<br/>apigateway-<br/>backend.parameters</b>            | Backend parameters.  |
| httpEndpoints     | No                | <b>x-<br/>apigateway-<br/>backend.http<br/>Endpoints</b>    | HTTP service definition.   |
| httpVpcEndpoints  | No                | <b>x-<br/>apigateway-<br/>backend.http<br/>VpcEndpoints</b> | HTTP-VPC service definition.   |
| functionEndpoints | No                | <b>x-<br/>apigateway-<br/>backend.functionEndpoints</b>     | Function service definition.   |
| mockEndpoints     | No                | <b>x-<br/>apigateway-<br/>backend.mockEndpoints</b>         | Mock service definition.   |

| Parameter  | Mandatory | Type                                     | Description               |
|------------|-----------|--|---------------------------|
| conditions | Yes       | x-apigateway-backend-policies.conditions | Backend policy condition. |

## 7.1: x-apigateway-backend-policies.conditions

**Meaning:** API backend policy.

**Scope of effect:** x-apigateway-backend-policies

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
          x-apigateway-request-type: "public"
          x-apigateway-backend:
            type: "backend endpoint type"
          x-apigateway-backend-policies:
            - type: "backend endpoint type"
              name: "backend policy name"
              conditions:
                - type: "equal/enum/pattern",
                  value: "string",
                  origin: "source/request_parameter",
                  parameter_name: "string"
```

**Table 6-51** Parameter description

| Parameter | Mandatory | Type   | Description   |
|-----------|-----------|--------|---|
| type      | Yes       | String | Policy condition type. <b>equal</b> , <b>enum</b> , and <b>pattern</b> are supported. |
| value     | Yes       | String | Policy condition value.   |
| origin    | Yes       | String | Policy condition source. <b>source</b> and <b>request</b> are supported.              |
| parameter | No        | String | Input parameter name if the <b>origin</b> parameter is set to <b>request</b> .        |

## 8: x-apigateway-ratelimit

**Meaning:** Request throttling policy.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-ratelimit: 'customRatelimitName'
```

**Table 6-52** Parameter description

| Parameter              | Mandatory | Type   | Description  |
|------------------------|-----------|--------|--|
| x-apigateway-ratelimit | No        | String | Name of the referenced request throttling policy. Set this parameter to <b>customRatelimitName</b> . |

## 9: x-apigateway-ratelimits

**Meaning:** Mapping between a request throttling policy name and limit values.

**Scope of effect:** [Swagger Object](#)

### Example

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: second/minute/hour
    shared: true
    special:
      - type: APP
        limit: 100
    instance: xxxxxxxx
```

**Table 6-53** Parameter description

| Parameter           | Mandatory | Type   | Description   |
|---------------------|-----------|--|---|
| customRatelimitName | No        | <a href="#">x-apigateway-ratelimits.policy</a> | Custom request throttling policy. To use a request throttling policy, set <a href="#">x-apigateway-ratelimit</a> to the name of the policy. |

### 9.1: x-apigateway-ratelimits.policy

**Meaning:** Definition of a request throttling policy.

**Scope of effect:** [x-apigateway-ratelimits](#)

**Example**

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: MINUTE
    shared: false
    special:
      - type: USER
        limit: 100
      instance: xxxxxxx
```

**Table 6-54** Parameter description

| Parameter  | Man<br>dator<br>y | Type   | Description   |
|------------|-------------------|--|---|
| api-limit  | Yes               | Number   | Maximum number of times an API can be called.   |
| user-limit | No                | Number   | Number of times the API can be called by a user.  |
| app-limit  | No                | Number   | Number of times the API can be called by an app.  |
| ip-limit   | No                | Number   | Number of times the API can be called by an IP address.                                     |
| interval   | Yes               | Number   | Throttling period.  |
| unit       | Yes               | String   | Throttling unit, which can be <b>SECOND</b> , <b>MINUTE</b> , <b>HOUR</b> , or <b>DAY</b> . |
| shared     | No                | Boolean  | Whether to share the throttling limits among APIs.  |
| special    | No                | <a href="#">x-apigateway-ratelimits.policy.special</a> | Special request throttling policy.  |

**9.2: x-apigateway-ratelimits.policy.special**

**Meaning:** Definition of a special request throttling policy.

**Scope of effect:** [x-apigateway-ratelimits.policy](#)

**Example**

```
x-apigateway-ratelimits:
  customRatelimitName:
```

```
api-limit: 200
app-limit: 200
user-limit: 200
ip-limit: 200
interval: 1
unit: MINUTE
shared: false
special:
- type: USER
  limit: 100
instance: xxxxxxxx
```

**Table 6-55** Parameter description

| Parameter | Mandator y | Type   | Description  |
|-----------|------------|--------|--|
| type      | Yes        | String | Excluded request throttling type, which can be <b>APP</b> or <b>USER</b> . |
| limit     | Yes        | Number | Access limit.  |
| instance  | Yes        | String | Excluded app or user   |

## 10: x-apigateway-access-control

**Meaning:** Access control policy.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-access-control: 'customAccessControlName'
```

**Table 6-56** Parameter description

| Parameter                   | Mandator y | Type   | Description  |
|-----------------------------|------------|--------|--|
| x-apigateway-access-control | No         | String | Name of the referenced access control policy. Set this parameter to <b>customAccessControlName</b> . |

## 11: x-apigateway-access-controls

**Meaning:** Mapping between an access control policy name and limit settings.

**Scope of effect:** [Swagger Object](#)

### Example

```
x-apigateway-access-controls:
  customAccessControlName:
```

```
acl-type: "DENY"
entity-type: "IP"
value: 127.0.0.1,192.168.0.1/16
```

**Table 6-57** Parameter description

| Parameter               | Mandatory | Type                                       | Description  |
|-------------------------|-----------|--|--|
| customAccessControlName | No        | <b>x-apigateway-access-controls.policy</b> | Custom access control policy. To use an access control policy, set <b>x-apigateway-access-control</b> to the name of the policy. |

## 11.1: x-apigateway-access-controls.policy

**Meaning:** Definition of an access control policy.

**Scope of effect:** **x-apigateway-access-controls**

### Example

```
x-apigateway-access-controls:
customAccessControlName:
acl-type: "DENY"
entity-type: "IP"
value: 127.0.0.1,192.168.0.1/16
```

**Table 6-58** Parameter description

| Parameter   | Mandatory | Type   | Description  |
|-------------|-----------|--------|--|
| acl-type    | Yes       | String | Access control effect. The options include <b>PERMIT</b> and <b>DENY</b> . |
| entity-type | Yes       | String | Access control object. Only IP addresses are supported.                    |
| value       | Yes       | String | Access control policy values. Use commas (,) to separate multiple values.  |

## 12: x-apigateway-roma-app

**Meaning:** Integration application bound to the API.

**Scope of effect:** **Operation Object**

### Example

```
paths:
'/path':
```



```
get:
  x-apigateway-roma-app: 'romaAppName'
```

**Table 6-59** Parameter description

| Parameter             | Mandatory | Type   | Description   |
|-----------------------|-----------|--------|---|
| x-apigateway-roma-app | Yes       | String | Name of the integration application bound to the API. |

## 13 x-apigateway-plugins

**Meaning:** API plug-in service.

**Scope of effect:** [Operation Object](#)

**Example:**

```
paths:
  '/path':
    get:
      x-apigateway-plugins: ['Plugin_mock']
```

**Table 6-60** Parameter description

| Parameter            | Mandatory | Type  | Description                        |
|----------------------|-----------|-------|------------------------------------|
| x-apigateway-plugins | No        | Array | List of plug-ins bound to the API. |

## 6.7 Managing Custom Backends

### 6.7.1 Taking a Custom Backend Offline

If a custom backend is not needed to provide services, take it offline. Its published frontend API will also be taken offline and deleted.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, take one or more backends offline.

When the backend status changes to **Developing**, the backend is taken offline.

**NOTICE**

Taking a backend offline will make the corresponding frontend API inaccessible. Before performing this operation, ensure that you have notified the users who have used this API.

3. In the navigation pane on the left, choose **API Connect** > **APIs** to check whether the API corresponding to the backend has been deleted.

## 6.7.2 Importing and Exporting Custom Backends

ROMA Connect allows you to import and export custom backends using YAML and JSON files. When importing APIs, file content must comply with the Swagger 2.0 specifications.

### Prerequisites

- Supplement the [Swagger extended definitions for custom backends](#) in the API file to import.
- Before importing a custom backend, ensure that there is sufficient custom backend quota.
- When importing a custom backend, the maximum size of a target API definition file is 3 MB.
- When exporting a custom backend, the maximum size of a target API definition file is 50 MB. If the size of the target API definition file exceeds 50 MB, the excess part will not be exported.

### Importing Custom Backends

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect** > **Custom Backends**. On the **Backends** tab page, click **Import** above the backend list.
3. On the **Import Backend** page, configure backend import information.

**Table 6-61** Parameters for importing custom backends

| Parameter                     | Description  |
|-------------------------------|--|
| Basic Definition Overwrite    | Specifies whether to overwrite the existing backends when the imported backends conflict with the existing ones.   |
| Extended Definition Overwrite | Specifies whether to overwrite the existing extended information, such as the request throttling policy and access control policy, when there is a conflict between the imported custom backends and ROMA Connect. <ul style="list-style-type: none"><li>• Enable: The existing extended information is overwritten with that defined in the imported file.</li><li>• Disable: The existing extended information is used, rather than that in the imported file.</li></ul> |

| Parameter | Description  |
|-----------|--|
| Swagger   | Click <b>Select</b> and select a local YAML or JSON file that complies with Swagger specifications.<br>After the file is imported, you can preview and modify the file online. |

4. Click **OK**.

The import result is displayed in the right pane. The **success** field contains the backends that are successfully imported, and the **failure** field contains the failed backends and their error codes and error description.

## Exporting Custom Backends

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, click **Export** above the backend list.  
Only backends in the **Developing** state can be exported.
3. On the **Export Backend** page, configure backend export information.

**Table 6-62** Parameters for exporting backends

| Parameter               | Description  |
|-------------------------|--|
| Scope                   | Specify the scope of custom backends to be exported. <ul style="list-style-type: none"><li>• <b>All</b>: All custom backends are exported. Only a random one of the custom backends with the same request path among different integration applications can be exported.</li><li>• <b>Current application</b>: Custom backends of an integration application are exported.</li></ul> |
| Integration Application | Mandatory for <b>Scope</b> set to <b>Current application</b> .<br>Select the integration application to which the custom backends to export belong.  |
| API Definition          | Specify the type of API definition to be exported. Only <b>Full</b> is supported, indicating that all requests and service information of backends are exported.   |
| Format                  | Select the format of the exported API file.<br>Options: <b>YAML</b> or <b>JSON</b>   |

4. Click **Export** to export the API file to a local directory. The content of the exported file is displayed in the right pane.

## 6.7.3 Adding Public Configurations for Custom Backends

Global public configuration items, such as variables, passwords, and certificates, can be added for a custom backend. This lets you quickly reference the added configuration items in the JavaScript script of a function backend.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Configurations** tab, click **Add Configuration**.
3. In the dialog box displayed, configure related information and click **OK**.

**Table 6-63** Adding public backend configurations

| Parameter               | Description   |
|-------------------------|---|
| Configuration Name      | Enter a configuration name.   |
| Integration Application | Select an integration application for the configuration item.   |
| Configuration Type      | Select a configuration type.<br>Options: <b>Template variable</b> , <b>Password</b> , <b>Certificate</b>  |
| Configuration Value     | Available for <b>Configuration Type</b> set to <b>Template variable</b> or <b>Password</b> .<br>Enter the template variable or password.  |
| Confirm Value           | Available for <b>Configuration Type</b> set to <b>Password</b> .<br>Enter the password again, which must be the same as the value of <b>Configuration Value</b> .   |
| Certificate             | Available for <b>Configuration Type</b> set to <b>Certificate</b> .<br>Enter the certificate in PEM format.<br>Open the PEM certificate file in the certificate to be uploaded in text, and copy the certificate content to <b>Certificate</b> .                |
| Private Key             | Available for <b>Configuration Type</b> set to <b>Certificate</b> .<br>Enter the private key of the certificate in PEM format.<br>Open the KEY/PEM private key file in the certificate to be uploaded in text, and copy the private key to <b>Private Key</b> . |
| Password                | Available for <b>Configuration Type</b> set to <b>Certificate</b> .<br>Enter the password of the certificate private key.   |

| Parameter        | Description  |
|------------------|--|
| Confirm Password | Available for <b>Configuration Type</b> set to <b>Certificate</b> .<br>Enter the password of the certificate private key again, which must be the same as the value of <b>Password</b> . |
| Description      | Enter a description of the configuration item.   |

- Reference the configuration item in a backend function script.  
For a configuration item named **example**, the reference format of each type of configuration is as follows:
  - Template variable: #{example}
  - Password: CipherUtils.getPlainCipherText("example")
  - Certificate: CipherUtils.getPlainCertificate("example")

## 6.7.4 Appendix: Swagger Extended Definitions for Custom Backends

### Overview

Building on the basic Swagger definitions, ROMA Connect extends the definitions of APIs to the authentication mode and function script definition. This section describes the extended definitions for custom backends.

#### 1: x-livedata-auth-type

**Meaning:** Swagger-based apiKey authentication format, which defines an authentication mode provided by the custom backend.

**Scope of effect:** [Security Scheme Object](#)

#### Example

```
securityDefinitions:
  customize-name-signature:
    type: "apiKey"
    name: "Authorization"
    in: "header"
    x-livedata-auth-type: "SIGNATURE"
    x-livedata-signature:
      key: "signatureKey"
      secret: "signatureSecret"
```

**Table 6-64** Parameter description

| Parameter | Man<br>dator<br>y | Type   | Description   |
|-----------|-------------------|--------|---|
| type      | Yes               | String | Authentication type. Only <b>apiKey</b> is supported. |

| Parameter                   | Man<br>dator<br>y | Type   | Description   |
|-----------------------------|-------------------|--------|---|
| name                        | Yes               | String | Name of the parameter used for authentication. Set this parameter to <b>Authorization</b> . |
| in                          | Yes               | String | Location of the parameter. Only <b>header</b> is supported.                                 |
| description                 | No                | String | Description of the parameter.   |
| x-livedata-auth-type        | Yes               | String | Custom backend authentication mode. The value can only be <b>SIGNATURE</b> .                |
| x-livedata-signature.key    | No                | String | Key required for signature.   |
| x-livedata-signature.secret | No                | String | Secret required for signature.  |

## 2: x-livedata-version

**Meaning:** version number of the custom backend.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-livedata-version: '1.0.1'
```

**Table 6-65** Parameter description

| Parameter          | Man<br>dator<br>y | Type   | Description  |
|--------------------|-------------------|--------|--------------|
| x-livedata-version | Yes               | String | API version. |

## 3: x-livedata-status

**Meaning:** Custom backend status.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
```

```
get:  
x-livedata-status: 'DESIGNED'
```

**Table 6-66** Parameter description

| Parameter         | Mandatory | Type   | Description   |
|-------------------|-----------|--------|---|
| x-livedata-status | Yes       | String | Status of the custom backend. The options are <b>DESIGNED</b> , <b>DEVELOPED</b> , <b>TESTED</b> , and <b>DEPLOYED</b> . <ul style="list-style-type: none"><li>• <b>DESIGNED</b>: The custom backend has been designed and is to be developed.</li><li>• <b>DEVELOPED</b>: The custom backend has been developed and is to be tested.</li><li>• <b>TESTED</b>: The custom backend has been tested and is to be deployed.</li><li>• <b>DEPLOYED</b>: The custom backend has been deployed.</li></ul> |

## 4: x-livedata-roma-app

**Meaning:** Integration application bound to the custom backend.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:  
  '/path':  
    get:  
      x-livedata-roma-app: 'romaAppName'
```

**Table 6-67** Parameter description

| Parameter           | Mandatory | Type   | Description  |
|---------------------|-----------|--------|--|
| x-livedata-roma-app | Yes       | String | Integration application bound to the custom backend. |

## 5: x-livedata-scripts

**Meaning:** custom backend definition script.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-livadata-scripts:
        - type: "function"
          content: "custom-script-content"
          result: "func"
```

**Table 6-68** Parameter description

| Parameter   | Man<br>dator<br>y | Type                                       | Description   |
|-------------|-------------------|--|---|
| content     | Yes               | String                                     | Script statement, which is a character string encoded using Base64. The actual script needs to be decoded using Base64.   |
| result      | Yes               | String                                     | Return object. The execution result of the statement will be encapsulated in the object and returned.<br><br>This parameter is valid only for data backends, not function backends. |
| type        | Yes               | String                                     | Script type. The options are <b>Function</b> , <b>SQL</b> , and <b>SP</b> .   |
| datasources | No                | <b>x-livadata-<br/>scripts.datasources</b> | Data source definition.   |

## 5.1 x-livadata-scripts.datasources

**Meaning:** data source definition of the custom backend.

**Scope of effect:** **x-livadata-scripts**

### Example

```
paths:
  '/users':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-livadata-scripts:
        - type: "function"
          content: "custom-script-content"
          result: "custom-script-result"
          datasource:
            name: "custom-datasource-name"
```



**Table 6-69** Parameter description

| Parameter | Mandatory | Type   | Description       |
|-----------|-----------|--------|-------------------|
| name      | Yes       | String | Data source name. |

## 6.8 Configuring API Control Policies

### 6.8.1 Configuring API Request Throttling

Request throttling limits the number of times an API can be called within a period to protect the backend service. To provide continuous and stable services, create request throttling policies. If request throttling is triggered for an API, all API calling requests during the request throttling period will be discarded and a failure response will be returned to the calling party.

A request throttling policy and an API are independent of each other. A request throttling policy takes effect for an API only after it is bound to the API.

#### Constraints

An API can be bound to only one request throttling policy in an environment, but each request throttling policy can be bound to multiple APIs.

#### Creating a Request Throttling Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Request Throttling** in the **Traditional Policy** area.
4. On the page displayed, configure request throttling information.

**Table 6-70** Parameters for creating a request throttling policy

| Parameter   | Description  |
|-------------|--|
| Policy Name | Enter a request throttling policy name. Using naming rules facilitates future search.  |
| Type        | Select the type of the request throttling policy. <ul style="list-style-type: none"><li>• <b>API-specific:</b> Requests are throttled based on each API the policy is bound to.</li><li>• <b>API-sharing:</b> Requests are throttled based on all APIs as a whole.</li></ul> |

| Parameter                | Description  |
|--------------------------|--|
| Period                   | Enter the request throttling duration in seconds, minutes, hours, or days. This parameter must be used together with request limit parameters: <ul style="list-style-type: none"><li>• <b>Max. API Requests</b> limits calls</li><li>• <b>Max. User Requests</b> limits calls by a user</li><li>• <b>Max. Credential Requests</b> limits calls by a credential</li><li>• <b>Max. IP Address Requests</b> limits calls by an IP address</li></ul> |
| Max. API Requests        | Enter the maximum number of times that an API can be called. This parameter is used along with <b>Period</b> .   |
| Max. User Requests       | Enter the maximum number of times that an API can be called by a user. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than the <b>Max. API Requests</b> .   |
| Max. Credential Requests | Enter the maximum number of times that an API can be called by a credential. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than the <b>Max. API Requests</b> .   |
| Max. IP Address Requests | Enter the maximum number of times that an API can be called by an IP address. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than the <b>Max. API Requests</b> .  |
| Description              | Describe the request throttling policy.  |

5. Click **OK**.

After the request throttling policy is created, perform the operations described in [Binding a Request Throttling Policy to an API](#) to make the policy take effect for the API.

## Binding a Request Throttling Policy to an API

1. On the **Policies** tab, filter policies by **Request Throttling**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to, and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Binding a Request Throttling Policy to an Application

To throttle requests for an integration application, add an excluded application to the request throttling policy. The **Max. Credential Requests** of the application will

then be restricted by the threshold of the excluded applications, while **Max. API Requests** and **Max. User Requests** are restricted by the throttling policy.

1. On the **Policies** tab, filter policies by **Request Throttling**.
2. Click the name of a request throttling policy.
3. Click the **Excluded Credentials** tab and click **Select Excluded App**.
4. In the **Select Excluded App** dialog box, configure application information.

**Table 6-71** Excluded app parameters

| Parameter | Description  |
|-----------|--|
| App       | Select the integration application type. <ul style="list-style-type: none"><li>• <b>Existing</b>: integration application created by the current user</li><li>• <b>Cross-tenant</b>: integration application created by another user. Enter the ID of the integration application created by another user in the current instance.</li></ul> |
| App Name  | This parameter is mandatory only if <b>App</b> is set to <b>Existing</b> .<br>Select the integration application the request throttling policy will bind to.   |
| Threshold | Enter the maximum number of times that an API can be called by the integration application within a specified period. The value of this parameter cannot be greater than the <b>Max. API Requests</b> in the request throttling policy.  |

5. Click **OK**.

## Binding a Request Throttling Policy to a Tenant

To throttle requests for a tenant, add an excluded tenant to the request throttling policy. The **Max. User Requests** of the tenant will then be limited by the threshold of the excluded tenant, while **Max. API Requests** and **Max. App Requests** are limited by the throttling policy.

1. On the **Policies** tab, filter policies by **Request Throttling**.
2. Click the name of a request throttling policy.
3. Click the **Excluded Tenants** tab and click **Select Excluded Tenant**.
4. In the **Select Excluded Tenant** dialog box, configure tenant information.

**Table 6-72** Parameters for configuring an excluded tenant

| Parameter  | Description   |
|------------|---|
| Account ID | <p>Enter the ID of the tenant the request throttling policy will bind to.</p> <ul style="list-style-type: none"><li>• If the App authentication mode is used to call APIs, the tenant ID is the project ID of the user to which the integration application belongs.</li><li>• If IAM authentication is used to call APIs, enter the account ID of the caller.</li></ul> <p>Click the username in the upper right corner of the console and choose <b>My Credentials</b> to obtain the project ID and account ID.</p> |
| Threshold  | <p>Enter the maximum number of times that an API can be called by the tenant within a specified period. The value of this parameter cannot be greater than the <b>Max. API Requests</b> in the request throttling policy.</p>   |

5. Click **OK**.

## 6.8.2 Configuring API Access Control

Access control protects backend services by controlling API access of IP addresses and accounts. Policies allow or deny the access of certain IP addresses or accounts to an API.

An access control policy and an API are independent of each other. An access control policy takes effect for an API only after it is bound to the API.

### Constraints

An API can be bound only to one access control policy of the same restriction type in an environment, but each access control policy can be bound to multiple APIs.

### Creating an Access Control Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Access Control** in the **Traditional Policy** area.
4. On the page displayed, configure access control information.

**Table 6-73** Parameters for creating an access control policy

| Parameter    | Description   |
|--------------|---|
| Policy Name  | Enter an access control policy name. Using naming rules facilitates future search.  |
| Type         | Select the restriction type of the access control policy. <ul style="list-style-type: none"><li>• <b>IP address</b>: restricts API calling by IP address.</li><li>• <b>Account name</b>: restricts API calling by account name. This option is available only to APIs using IAM authentication. The restriction also applies to the IAM users under the specified accounts. IAM users cannot be specified separately.</li><li>• <b>Account ID</b>: restricts API calling by account ID. This option is available only to APIs using IAM authentication. The restriction also applies to the IAM users under the specified accounts. IAM users cannot be specified separately.</li></ul> |
| Effect       | Select the access control type. This parameter is used along with <b>Restriction Type</b> . <ul style="list-style-type: none"><li>• <b>Allow</b>: Only specified IP addresses or accounts are allowed to call APIs.</li><li>• <b>Deny</b>: Specified IP addresses or accounts are not allowed to call APIs.</li></ul>   |
| IP Addresses | Mandatory for <b>Type</b> set to <b>IP address</b> .<br>Click <b>Add IP Address</b> to add the IP addresses or IP address segments that are allowed or forbidden to call an API.  |
| Account Name | Mandatory when <b>Type</b> is set to <b>Account name</b> .<br>Enter the account names that are allowed or forbidden to call an API. Use commas (,) to separate multiple account names.<br>Click the username in the upper right corner of the console. Choose <b>My Credentials</b> and obtain the account name on the <b>API Credentials</b> page.   |
| Account ID   | Mandatory when <b>Type</b> is set to <b>Account ID</b> .<br>Enter the account IDs that are allowed or forbidden to call an API. Use commas (,) to separate multiple account IDs.<br>Click the username in the upper right corner of the console. Choose <b>My Credentials</b> and obtain the account ID on the <b>API Credentials</b> page.   |

5. Click **OK**.

After the access control policy is created, you also need to perform the operations described in [Binding an Access Control Policy to an API](#) to make the policy take effect for the API.

## Binding an Access Control Policy to an API

1. On the **Policies** tab, filter policies by **Access Control**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

### 6.8.3 Configuring API Credential Quotas

A credential quota limits the total number of times that an API can be called by a credential in a specified period to protect backend services. You can create a credential quota policy to limit the number of API calls made by credentials bound to the policy.

A credential quota policy and a client are independent of each other. The policy takes effect on the credential only after the credential is bound to the policy.

#### Constraints

- One credential can be bound to only one credential quota policy, but one credential quota policy can be bound to multiple credentials.
- Only users assigned with the **Tenant Administrator** permission can view and configure credential quota policies.

#### Creating a Credential Quota Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Credentials**. On the **Credential Quota Policies** tab, click **Create Credential Quota Policy**.
3. On the page displayed, configure policy information.

**Table 6-74** Parameters for creating a credential quota policy

| Parameter | Description   |
|-----------|---|
| Name      | Enter a credential quota policy name. Using naming rules facilitates future search. |

| Parameter         | Description   |
|-------------------|---|
| Effective On      | Select the start time for the quota policy to take effect. For example, if <b>Effective On</b> is set to <b>Aug 8, 2020 05:05:00</b> and <b>Period</b> is set to 1 hour, the quota policy took effect on Aug 8, 2020 05:05:00, and the period from the fifth minute of an hour to the fifth minute of the next hour is a cycle, for example, from 05:05:00 to 06:05:00. |
| Period            | Enter the quota duration in seconds, minutes, hours, or days. This parameter must be used along with <b>Max. API Requests</b> to limit the total number of times an API can be called by a credential within the specified period.  |
| Max. API Requests | Enter the maximum number of times that an API can be called by a credential. This parameter is used along with <b>Period</b> .  |
| Description       | Describe the credential quota policy.   |

4. Click **OK**.

After the credential quota policy is created, you also need to perform the operation described in [Binding a Quota Policy to a Credential](#) to make the policy take effect for APIs.

## Binding a Quota Policy to a Credential

1. On the **Credential Quota Policies** tab page, click the name of a policy to go to the details page.
2. In the **Bound Credential List** area, click **Bind to Credentials**.
3. In the dialog box displayed, select the credentials to bind the policy to. Search by credential name is supported.
4. Click **OK**.

A credential can be bound only to one quota policy. If a credential is bound to a quota policy repeatedly, the original quota policy will be unbound.

## 6.8.4 Configuring API Credential Access Control

Credential access control protects backend services by controlling API access of credential IP addresses and accounts. You can configure an access policy to allow or forbid a credential with the specified IP address to access an API.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > Credentials**. On the **Credentials** tab page, click **Set Access Control** of a credential.
3. On the page displayed, configure credential access control information.

**Table 6-75** Parameters for creating an access control policy

| Parameter    | Description   |
|--------------|---|
| Effect       | Access control type. Options: <ul style="list-style-type: none"><li>• <b>Allow</b>: Only credentials with specified IP addresses are allowed to call APIs.</li><li>• <b>Deny</b>: Credentials with specified IP addresses are not allowed to call APIs.</li></ul> |
| IP Addresses | Click <b>Add IP Address</b> to add the credential IP addresses or IP address segments that are allowed or forbidden to call an API.   |

4. Click **OK**.

## 6.9 Configuring API Plug-in Policies

### 6.9.1 Configuring CORS

For security, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send cross-domain **XMLHttpRequest** requests. For details about CORS, see [Configuring CORS for APIs](#).

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. The CORS policy enables the specifying of preflight request/response headers and the automatic creation of preflight request APIs for cross-domain API access.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in takes effect for the API only in the specified environment.

#### Constraints

- An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.
- In the same API group, all APIs published in the same environment and with the same request path can be bound only to the same CORS policy.
- If you have enabled CORS for an API and have also bound a CORS policy to the API, the policy will be used.
- If a request path contains an API with the **OPTIONS** method, none of the APIs in the request path can be bound to the CORS policy in the environment where the API is published.
- When you bind a plug-in policy to an API, ensure that the request method of the API is specified in **allow\_methods**.



## Creating a CORS Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **CORS** in the **Plug-ins** area.
4. On the page displayed, configure plug-in policy information.

**Table 6-76** Policy configuration

| Parameter  | Description   |
|--|---|
| Name   | Enter a policy name. Using naming rules facilitates future search.  |
| Type   | Fixed as <b>CORS</b> .  |
| Scope  | Specify the scope to view the policy. <ul style="list-style-type: none"> <li>• <b>Integration application:</b> Each policy belongs to an integration application. Only users who have the permission on the integration application can view and use the policy.</li> <li>• <b>All:</b> All users in the current instance can view and use the policy.</li> </ul> |
| Integration Application  | Mandatory for <b>Scope</b> set to <b>Integration application</b> . Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Description  | Describe the policy.  |
| <b>Policy Content:</b> Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> . |   |
| Allowed Origins  | Access-Control-Allow-Origin response header, which specifies the external domain URIs that are allowed to access the API. Use commas (,) to separate multiple URIs.<br><br>For requests that do not carry identity credentials, set this parameter to * to allow access requests from all domains.  |
| Allowed Methods  | The <b>Access-Control-Allow-Methods</b> header specifies the HTTP methods that can be used in a request.  |

| Parameter         | Description   |
|-------------------|---|
| Allowed Headers   | <p><b>Access-Control-Allow-Headers</b> response header, which specifies request headers that can be used when sending <b>XMLHttpRequest</b> requests. Use commas (,) to separate multiple headers.</p> <p>By default, simple request headers <b>Accept</b>, <b>Accept-Language</b>, <b>Content-Language</b>, and <b>Content-Type</b> (only if the value is <b>application/x-www-form-urlencoded</b>, <b>multipart/form-data</b>, or <b>text/plain</b>) are carried in requests. You do not need to configure these headers in this parameter.</p> |
| Exposed Headers   | <p><b>Access-Control-Expose-Headers</b> response header, which specifies which response headers can be contained in the response of <b>XMLHttpRequest</b>. Use commas (,) to separate multiple headers.</p> <p>By default, basic response headers <b>Cache-Control</b>, <b>Content-Language</b>, <b>Content-Type</b>, <b>Expires</b>, <b>Last-Modified</b>, and <b>Pragma</b> can be contained in the response. You do not need to configure these headers in this parameter.</p>   |
| Maximum Age       | <p><b>Access-Control-Max-Age</b> response header, which specifies the validity period (in seconds) of the preflight request. No more preflight requests are needed within the period.</p>   |
| Allow Credentials | <p>The <b>Access-Control-Allow-Credentials</b> header indicates whether the response to the request can be exposed to browsers.</p>   |

5. Click **OK**.

After a plug-in policy is created, perform [Binding a Plug-in Policy to an API](#) for the policy to take effect for the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab, filter policies by **CORS**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{
  "allow_origin": "**",
  "allow_methods": "GET,POST,PUT",
  "allow_headers": "Accept-Ranges,Cache-Control",
```

```
"expose_headers": "X-Request-Id,X-Apig-Latency",  
"max_age": 172800,  
"allow_credentials": true  
}
```

## 6.9.2 Configuring HTTP Response Header Management

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. An HTTP response header management policy enables you to customize HTTP response headers that will be returned in an API response.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in policy takes effect for the API only in the specified environment.

### Constraints

- An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.
- The system response headers (such as **x-apig-\*** and **x-request-id**) cannot be modified.
- The response headers for CORS cannot be modified.

### Creating an HTTP Response Header Management Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **HTTP Response Header Management** in the **Plug-ins** area.
4. On the page displayed, configure plug-in policy information.

**Table 6-77** Policy configuration

| Parameter | Description  |
|-----------|--|
| Name      | Enter a policy name. Using naming rules facilitates future search.   |
| Type      | Fixed as <b>HTTP Response Header Management</b> .  |
| Scope     | Specify the scope to view the policy. <ul style="list-style-type: none"><li>• <b>Integration application:</b> Each policy belongs to an integration application. Only users who have the permission on the integration application can view and use the policy.</li><li>• <b>All:</b> All users in the current instance can view and use the policy.</li></ul> |

| Parameter  | Description   |
|--|---|
| Integration Application  | Mandatory for <b>Scope</b> set to <b>Integration application</b> .<br>Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one. |
| Description  | Describe the policy.  |
| <b>Policy Content:</b> Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> . |   |

| Parameter        | Description  |
|------------------|--|
| Response Headers | <p>Click <b>Add Response Header</b> to add a response header.</p> <ul style="list-style-type: none"><li>• <b>Name:</b> name of the response header, which is case-insensitive and must be unique within a plug-in.</li><li>• <b>Value:</b> value of the response header. This parameter does not take effect and is left blank if you set <b>Action</b> to <b>Delete</b>.</li><li>• <b>Action:</b> operation of the response header. You can override, append, delete, skip, or add the specified header.<ul style="list-style-type: none"><li>- <b>Override</b><ul style="list-style-type: none"><li>- The value of this response header will override that of the same header that exists in an API response.</li></ul></li><li>- If an API response contains multiple headers with the same name as the one you set here, only the value of the specified header will be returned.<ul style="list-style-type: none"><li>- If an API response does not contain the specified header, the value you set here will be returned.</li></ul></li><li>- <b>Append</b><ul style="list-style-type: none"><li>- If an API response contains the specified header, the value you set here will be added, following the existing value. The two values will be separated with commas (,).</li><li>- If an API response contains multiple headers with the same name as the one you set here, values of these headers will be separated with commas (,) and followed by the value of the specified header.</li><li>- If an API response does not contain the specified header, the value you set here will be returned.</li></ul></li><li>- <b>Delete</b><ul style="list-style-type: none"><li>- If an API response contains the specified header, the header will be deleted.</li><li>- If an API response contains multiple headers with the same name as the one you set here, all these headers will be deleted.</li></ul></li><li>- <b>Skip</b><ul style="list-style-type: none"><li>- If an API response contains the specified header, the header will be skipped.</li><li>- If an API response contains multiple headers with the same name as the one you set here, values of all these headers will be returned without modification.</li><li>- If an API response does not contain the specified header, the value you set here will be returned.</li></ul></li></ul></li></ul> |

| Parameter | Description   |
|-----------|---|
|           | <ul style="list-style-type: none"><li>- <b>Add</b><br/>The value of the specified header will be returned even if the header does not exist in an API response.</li></ul> |

5. Click **OK**.

After a plug-in policy is created, perform [Binding a Plug-in Policy to an API](#) for the policy to take effect for the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab, filter policies by **HTTP Response Header Management**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{
  "response_headers": [
    {
      "name": "header1",
      "value": "test",
      "action": "append"
    },
    {
      "name": "header2",
      "value": "roma",
      "action": "override"
    }
  ]
}
```

### 6.9.3 Configuring Request Throttling 2.0

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. Request throttling 2.0 limits the number of times an API can be called within a specific time period to protect backend services.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in policy takes effect for the API only in the specified environment.

## Constraints

- An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.

- If an API has been bound by a traditional request throttling policy and a request throttling 2.0 policy at the same time, the request throttling 2.0 policy takes effect.

## Creating a Request Throttling 2.0 Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Request Throttling 2.0** in the **Plugins** area.
4. On the page displayed, configure plug-in policy information.


**Table 6-78** Policy configuration

| Parameter   | Description  |
|---|--|
| Name  | Enter a policy name. Using naming rules facilitates future search.   |
| Type  | Fixed as <b>Request Throttling 2.0</b> .   |
| Scope   | Specify the scope to view the policy. <ul style="list-style-type: none"><li>• <b>Integration application</b>: Each policy belongs to an integration application. Only users who have the permission on the integration application can view and use the policy.</li><li>• <b>All</b>: All users in the current instance can view and use the policy.</li></ul> |
| Integration Application   | Mandatory for <b>Scope</b> set to <b>Integration application</b> . Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| Description   | Describe the policy.   |
| <b>Policy Content</b> : Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> . |  |

| Parameter                 | Description  |
|---------------------------|--|
| Throttling                | <p>Set the throttling type. High-performance throttling is recommended.</p> <ul style="list-style-type: none"><li>• <b>High precision:</b> There is no error in a specific period, with poor performance in high concurrency scenarios. Select this type in low concurrency scenarios.</li><li>• <b>High-performance:</b> There are occasionally small errors in a specific period, with small performance penalty in high concurrency scenarios. Select this type in high concurrency scenarios.</li><li>• <b>Single-node:</b> There are certain errors in a specific period, with smaller performance penalty in high concurrency scenarios. Select this type in higher concurrency scenarios.</li></ul> |
| <b>Policy Information</b> |  |
| Policy Type               | <p>Select a policy type.</p> <ul style="list-style-type: none"><li>• <b>API-specific:</b> Requests of APIs bound to the current plug-in are calculated separately.</li><li>• <b>API-sharing:</b> Requests of all APIs bound to the current plug-in are calculated together.</li></ul>  |
| Period                    | <p>Enter the request throttling duration in seconds, minutes, hours, or days. This parameter must be used together with parameters in <b>Basic Throttling</b>:</p> <ul style="list-style-type: none"><li>• <b>Max. API Requests</b> limits calls</li><li>• <b>Max. User Requests</b> limits calls by a user</li><li>• <b>Max. Credential Requests</b> limits calls by a credential</li><li>• <b>Max. IP Address Requests</b> limits calls by an IP address</li></ul>   |
| <b>Basic Throttling</b>   |  |
| Max. API Requests         | <p>Enter the maximum number of times that an API can be called. This parameter is used along with <b>Period</b>.</p>   |
| Max. User Requests        | <p>Enter the maximum number of times that an API can be called by a user. This parameter is used along with <b>Period</b>. The value of this parameter cannot be greater than the <b>Max. API Requests</b>.</p>  |
| Max. Credential Requests  | <p>Enter the maximum number of times that an API can be called by a credential. This parameter is used along with <b>Period</b>. The value of this parameter cannot be greater than the <b>Max. API Requests</b>.</p>  |
| Max. IP Address Requests  | <p>Enter the maximum number of times that an API can be called by an IP address. This parameter is used along with <b>Period</b>. The value of this parameter cannot be greater than the <b>Max. API Requests</b>.</p>   |



| Parameter                  | Description  |
|----------------------------|--|
| Parameter-based Throttling | <p>Specify whether to enable parameter-based throttling. Once enabled, rules and throttling limits in the policy you configure here override those in <b>Basic Throttling</b>.</p> <ul style="list-style-type: none"> <li>• If a parameter-based throttling policy is matched, its throttling limits take effect.</li> <li>• If no parameter-based throttling policy is matched, throttling limits configured in <b>Basic Throttling</b> take effect.</li> </ul>   |
| Define Parameters          | <p>Define the parameters for rule matching. Click <b>Add Parameter</b> to add rule parameters.</p> <ul style="list-style-type: none"> <li>• <b>Parameter Location:</b> position of a parameter in an API request. <ul style="list-style-type: none"> <li>– <b>path:</b> API request URI. This parameter is configured by default.</li> <li>– <b>method:</b> API request method. This parameter is configured by default.</li> <li>– <b>header:</b> first value of the header parameter in an API request.</li> <li>– <b>query:</b> first value of the query parameter in an API request.</li> <li>– <b>system:</b> a system parameter.</li> </ul> </li> <li>• <b>Parameter:</b> name of a parameter used for rule matching.</li> </ul> |

| Parameter           | Description   |
|---------------------|---|
| Define Rules        | <p>Define the rules for parameter-based throttling. Click <b>Add Rule</b> to add rules. The system matches rules from top to bottom.</p> <ul style="list-style-type: none"> <li>• <b>Rules:</b> Click  to modify the condition expression. If there are three or more expressions, you can layer them by clicking <b>Set Lower Level</b>. <ul style="list-style-type: none"> <li>- =: equal to</li> <li>- !=: not equal to</li> <li>- <b>pattern:</b> regular expression</li> <li>- <b>enum:</b> enumerated values, separated by comma (,)</li> </ul> </li> <li>• <b>Max. API Requests:</b> Enter the maximum number of times that bound APIs can be called. This parameter is used along with <b>Period</b>.</li> <li>• <b>Period:</b> Enter the throttling duration in seconds, minutes, hours, or days. This parameter is used along with <b>Max. API Requests</b>.</li> </ul> <p>For example, add the <b>Host</b> parameter and specify the location as <b>header</b>; add the condition <b>Host = www.abc.com</b>, and set the throttling limit to <b>10</b> and the period to 60s. For APIs whose <b>Host</b> parameter in the request header is equal to <b>www.abc.com</b>, they cannot be called again once called 10 times in 60s.</p> |
| Excluded Throttling | <p>Specify whether to enable excluded throttling for tenants or integration applications.</p> <p>The throttling limits for excluded tenants and applications override the <b>Max. User Requests</b> and <b>Max. Credential Requests</b> set in <b>Basic Throttling</b>.</p>   |

| Parameter        | Description  |
|------------------|--|
| Excluded Tenants | <p>Click <b>Add Tenant</b> to limit the requests of specified tenants.</p> <ul style="list-style-type: none"><li>• <b>Account ID:</b> Enter the ID of the tenant to which the request throttling policy is to be bound.<ul style="list-style-type: none"><li>- If the App authentication mode is used to call APIs, the tenant ID is the project ID of the user to which the integration application belongs.</li><li>- If IAM authentication is used to call APIs, enter the account ID of the caller.</li></ul></li></ul> <p>Click the username in the upper right corner of the console and choose <b>My Credentials</b> to obtain the project ID and account ID.</p> <ul style="list-style-type: none"><li>• <b>Threshold:</b> Enter the maximum number of times that an API can be called by the tenant within the specified period. The value of this parameter cannot be greater than the <b>Max. API Requests in Basic Throttling</b>.</li></ul> |
| Excluded Apps    | <p>Click <b>Add App</b> to limit the requests of specified integration applications.</p> <ul style="list-style-type: none"><li>• <b>App:</b> Select an integration application for request throttling.</li><li>• <b>Threshold:</b> Enter the maximum number of times that an API can be called by the application within the specified period. The value of this parameter cannot be greater than the <b>Max. API Requests in Basic Throttling</b>.</li></ul>  |

5. Click **OK**.

After a plug-in policy is created, perform [Binding a Plug-in Policy to an API](#) for the policy to take effect for the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab, filter policies by **Request Throttling 2.0**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to, and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{  
  "scope": "basic",  
  "default_interval": 60,  
  "default_time_unit": "second",  
}
```

```
"api_limit": 100,
"app_limit": 50,
"user_limit": 50,
"ip_limit": 20,
"specials": [
  {
    "type": "app",
    "policies": [
      {
        "key": "e9230d70c749408eb3d1e838850cdd23",
        "limit": 10
      }
    ]
  },
  {
    "type": "user",
    "policies": [
      {
        "key": "878f1b87f71c40a7a15db0998f358bb9",
        "limit": 10
      }
    ]
  }
],
"algorithm": "counter",
"parameters": [
  {
    "id": "3wuj354lpptv0toe0",
    "value": "reqPath",
    "type": "path",
    "name": "reqPath"
  },
  {
    "id": "53h7e7j11u38l3ocp",
    "value": "method",
    "type": "method",
    "name": "method"
  },
  {
    "id": "vv502bnb6g40td8u0",
    "value": "Host",
    "type": "header",
    "name": "Host"
  }
],
"rules": [
  {
    "match_regex": "[\"Host\", \"==\", \"www.abc.com\"]",
    "rule_name": "u8mb",
    "time_unit": "second",
    "interval": 2,
    "limit": 5
  }
]
}
```

## 6.9.4 Configuring Kafka Log Push

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. A Kafka log push policy pushes API call logs to Kafka so that users can easily obtain them.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in policy takes effect for the API only in the specified environment.

## Constraints

- An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.
- A maximum of five Kafka log push policies can be created on a ROMA Connect instance.
- In the log data to be pushed, the response data does not support **Transfer Encoding** response header parameters.
- By default, a maximum of 4 KB of logs can be pushed. Excess logs will be truncated.
- The request body and response body in pushed logs are calculated in UTF-8 bytes.

## Creating a Kafka Log Push Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Kafka Log Push** in the **Plug-ins** area.
4. On the page displayed, configure plug-in policy information.

**Table 6-79** Policy configuration

| Parameter   | Description  |
|---|--|
| Name  | Enter a policy name. Using naming rules facilitates future search.   |
| Type  | Fixed as <b>Kafka Log Push</b> .   |
| Scope   | Specify the scope to view the policy. <ul style="list-style-type: none"><li>• <b>Integration application</b>: Each policy belongs to an integration application. Only users who have the permission on the integration application can view and use the policy.</li><li>• <b>All</b>: All users in the current instance can view and use the policy.</li></ul> |
| Integration Application   | Mandatory for <b>Scope</b> set to <b>Integration application</b> . Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| Description   | Describe the policy.   |
| <b>Policy Content</b> : Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> . |  |
| <b>Policy Information</b>   |  |

| Parameter                     | Description  |
|-------------------------------|--|
| Broker Addresses              | Connection addresses of target Kafka brokers to which logs are to be pushed. Use commas (,) to separate multiple addresses.  |
| Topic                         | Topic of the target Kafka to report logs to.   |
| Key                           | Enter the key value of a message so that the message will be stored in a specified partition of Kafka. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner.   |
| Retry                         | <p>Configuration for retrying when logs fail to be pushed to Kafka.</p> <ul style="list-style-type: none"> <li>• <b>Retry Times:</b> the number of retry attempts in case of a failure<br/>Range: 0 to 5</li> <li>• <b>Retry Interval:</b> the interval of retry attempts in case of a failure, in seconds<br/>Range: 1 to 10</li> </ul> |
| <b>SASL Configuration</b>     |  |
| Security Protocol             | <p>Protocol used for connecting to the target Kafka.</p> <ul style="list-style-type: none"> <li>• <b>PLAINTEXT:</b> user authentication protocol of the default access point.</li> <li>• <b>SASL_PLAINTEXT:</b> SASL user authentication protocol.</li> <li>• <b>SASL_SSL:</b> SSL user authentication protocol.</li> </ul>              |
| Message Tx/Rx Mechanism       | Message transmission and receiving mechanism of the target Kafka. The default value is <b>PLAIN</b> .  |
| SASL Username                 | <p>Mandatory for <b>Security Protocol</b> set to <b>SASL_PLAINTEXT</b> or <b>SASL_SSL</b>.</p> <p>Username used for SASL or SSL authentication.</p>  |
| SASL Password                 | <p>Mandatory for <b>Security Protocol</b> set to <b>SASL_PLAINTEXT</b> or <b>SASL_SSL</b>.</p> <p>Password used for SASL or SSL authentication.</p>  |
| Confirm SASL Password         | <p>Mandatory for <b>Security Protocol</b> set to <b>SASL_PLAINTEXT</b> or <b>SASL_SSL</b>.</p> <p>Enter the SASL or SSL authentication password again.</p>   |
| Certificate Content           | <p>Available for <b>Security Protocol</b> set to <b>SASL_SSL</b>.</p> <p>CA certificate used for SSL authentication.</p>   |
| <b>Metadata Configuration</b> |  |

| Parameter             | Description  |
|-----------------------|--|
| System Metadata       | System fields that need to be included in pushed logs. The following fields are selected by default: <b>start_time</b> , <b>request_id</b> , <b>client_ip</b> , <b>request_time</b> , <b>http_status</b> , <b>scheme</b> , <b>request_method</b> , <b>upstream_addr</b> , <b>upstream_status</b> , <b>upstream_response_time</b> , <b>http_x_forwarded_for</b> , <b>http_user_agent</b> , and <b>error_type</b> . You can also specify other system fields.  |
| Request Data          | API request information that needs to be included in pushed logs. <ul style="list-style-type: none"><li>• <b>The log contains the request header:</b> Specify a header that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li><li>• <b>The log contains the request QueryString:</b> Specify a query string that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li><li>• <b>The log contains the request body:</b> If this option is selected, logs will contain the body of API requests.</li></ul> |
| Response Data         | API response information that needs to be included in pushed logs. <ul style="list-style-type: none"><li>• <b>The log contains the response header:</b> Specify a header that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li><li>• <b>The log contains the response body:</b> If this option is selected, logs will contain the body of API request responses.</li></ul>  |
| Custom Authentication | Custom authentication information that needs to be included in pushed logs. <ul style="list-style-type: none"><li>• <b>Frontend:</b> Enter a response field of frontend authentication that needs to be included. Separate multiple fields with commas (,).</li><li>• <b>Backend:</b> Enter a response field of backend authentication that needs to be included. Separate multiple fields with commas (,).</li></ul>  |

5. Click **OK**.

After a plug-in policy is created, perform [Binding a Plug-in Policy to an API](#) for the policy to take effect for the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab, filter policies by **Kafka Log Push**.
2. Click the name of a policy to go to the details page.

3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{
  "broker_list": [
    "10.10.10.10:81",
    "10.10.10.11:82",
    "10.10.10.12:83"
  ],
  "topic": "Topic-test",
  "key": "aaa",
  "max_retry_count": 0,
  "retry_backoff": 1,
  "sasl_config": {
    "security_protocol": "PLAINTEXT",
    "sasl_mechanisms": "PLAIN",
    "sasl_username": "",
    "sasl_password": "",
    "ssl_ca_content": ""
  },
  "meta_config": {
    "system": {
      "start_time": true,
      "request_id": true,
      "client_ip": true,
      "api_id": false,
      "user_name": false,
      "app_id": false,
      "request_time": true,
      "http_status": true,
      "server_protocol": false,
      "scheme": true,
      "request_method": true,
      "host": false,
      "api_uri_mode": false,
      "uri": false,
      "request_size": false,
      "response_size": false,
      "upstream_uri": false,
      "upstream_addr": true,
      "upstream_status": true,
      "upstream_connect_time": false,
      "upstream_header_time": false,
      "upstream_response_time": true,
      "http_x_forwarded_for": true,
      "http_user_agent": true,
      "error_type": true,
      "access_model1": false,
      "access_model2": false,
      "inner_time": false,
      "proxy_protocol_vni": false,
      "proxy_protocol_vpce_id": false,
      "proxy_protocol_addr": false,
      "body_bytes_sent": false,
      "api_name": false,
      "app_name": false,
      "provider_app_id": false,
      "all_upstream_response_time": false,
      "region_id": false,
      "auth_type": false,
      "response_source": false,
      "provider_app_name": false,
    }
  }
}
```



```
"custom_data_log01": false,
"custom_data_log02": false,
"custom_data_log03": false,
"custom_data_log04": false,
"custom_data_log05": false,
"custom_data_log06": false,
"custom_data_log07": false,
"custom_data_log08": false,
"custom_data_log09": false,
"custom_data_log10": false
},
"call_data": {
  "log_request_header": true,
  "request_header_filter": "aaa",
  "log_request_query_string": true,
  "request_query_string_filter": "bbb",
  "log_request_body": true,
  "log_response_header": true,
  "response_header_filter": "ccc",
  "log_response_body": true,
  "custom_authorizer": {
    "frontend": [],
    "backend": []
  }
}
}
```

## 6.9.5 Configuring Circuit Breakers

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. Circuit breakers are provided to protect the system when performance issues occur on backend services. When the backend service of an API times out for  $N$  consecutive times or there is long latency, the circuit breaker downgrade mechanism is triggered to return an error to the API caller or forward the request to the specified backend. After the backend service recovers, the circuit breaker closes and the request becomes normal.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in policy takes effect for the API only in the specified environment.

### Constraints

An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.

### Creating a Circuit Breaker Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Circuit Breaker** in the **Plug-ins** area.
4. On the page displayed, configure plug-in policy information.

**Table 6-80** Policy configuration


| Parameter  | Description   |
|--|---|
| Name   | Enter a policy name. Using naming rules facilitates future search.  |
| Type   | Fixed as <b>Circuit Breaker</b> .   |
| Scope  | Specify the scope to view the policy. <ul style="list-style-type: none"><li>• <b>Integration application:</b> Each policy belongs to an integration application. Only users who have the permission on the integration application can view and use the policy.</li><li>• <b>All:</b> All users in the current instance can view and use the policy.</li></ul>  |
| Integration Application  | Mandatory for <b>Scope</b> set to <b>Integration application</b> . Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Description  | Describe the policy.  |
| <b>Policy Content:</b> Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> . |   |
| Policy Type  | Select a policy type. <ul style="list-style-type: none"><li>• <b>API-specific:</b> APIs bound to the current plug-in are calculated separately for circuit breaker triggering.</li><li>• <b>API-sharing:</b> All APIs bound to the current plug-in are calculated together for circuit breaker triggering.</li></ul>  |
| Trigger Configuration  |   |
| Circuit Breaker Type   | Specify a circuit breaker triggering type. <ul style="list-style-type: none"><li>• <b>Timeout downgrade:</b> The circuit breaker is triggered upon backend timeout.</li><li>• <b>Conditional downgrade:</b> The circuit breaker is triggered whenever conditions configured in <b>Match Conditions</b> are met.</li></ul>   |
| Condition Type   | Specify a circuit breaker triggering mode. <ul style="list-style-type: none"><li>• <b>Count:</b> Once the number of requests that meet the conditions specified within the time window reaches the threshold, the circuit breaker is immediately triggered.</li><li>• <b>Percentage:</b> Once the percentage of requests that meet the conditions specified within the time window reaches the threshold, the circuit breaker is triggered at the end of the time window.</li></ul> |

| Parameter        | Description   |
|------------------|---|
| Match Conditions | <p>This parameter is mandatory only when <b>Circuit Breaker Type</b> is set to <b>Conditional downgrade</b>.</p> <p>Configure the match conditions for circuit breaker triggering.</p> <ul style="list-style-type: none"><li>• <b>Response Error Codes:</b> The circuit breaker is triggered when the backend responds with the status codes specified.</li><li>• <b>Response Latency:</b> The circuit breaker is triggered when the backend response latency exceeds the threshold specified.</li></ul>  |
| Time Window (s)  | <p>Configure the time window for circuit breaker triggering. Use this parameter together with <b>Threshold</b> or <b>Min Percentage</b>. Once the number of times or percentage when conditions are met within the time window exceeds the threshold, the circuit breaker is triggered.</p>   |
| Threshold        | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Count</b>.</p> <p>Configure the threshold for circuit breaker triggering. Use this parameter together with <b>Time Window (s)</b>. Once the number of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered.</p> <p><b>NOTE</b></p> <p>A circuit breaker plug-in is triggered for a single gateway component. If ROMA Connect has multiple components, the triggering for each component is determined separately.</p> <p>If the threshold is reached within the time window for a gateway component, requests sent to the component trigger the circuit breaker, but other gateway components still forward requests normally.</p> <p>An IP address indicates a gateway component. To view the number of gateway components, go to the <b>Instance Information</b> page of a ROMA Connect instance by clicking the instance name on the console and view the number of IP addresses in <b>Connection Addresses</b>.</p> |
| Min Calls        | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Percentage</b>.</p> <p>Set the minimum number of API calls that will trigger the circuit breaker within the time period. The circuit breaker will not be triggered if the number of API calls within the time period is less than this value.</p>   |

| Parameter                | Description  |
|--------------------------|--|
| Min Percentage (%)       | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Percentage</b>.</p> <p>Configure the threshold for circuit breaker triggering. Use this parameter together with <b>Time Window (s)</b>. Once the percentage of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered.</p>   |
| Control Duration (s)     | <p>Configure the duration for which the circuit breaker is open. The circuit breaker is closed when the duration reaches the value specified.</p>  |
| Backend Downgrade Policy | <p>Specify whether to enable the backend downgrade policy.</p> <ul style="list-style-type: none"><li>• Enable: Requests for APIs that have triggered a downgrade will be forwarded to a specified backend.</li><li>• Disable: Requests for APIs that have triggered a downgrade will not be forwarded to any backend. Instead, an error message indicating that the service is unavailable will be returned.</li></ul> |

| Parameter           | Description   |
|---------------------|---|
| Backend Policy Type | <p>This parameter is mandatory only when <b>Backend Downgrade Policy</b> is enabled.</p> <p>Specify the backend type to which requests will be forwarded when the circuit breaker is open.</p> <ul style="list-style-type: none"><li>● <b>Mock</b>: The defined response will be returned.<ul style="list-style-type: none"><li>- <b>Status Code</b>: the status code to be included in the response</li><li>- <b>Response</b>: the response body, which is in JSON format</li><li>- <b>Response Header</b>: header parameters to be included in the response</li></ul></li><li>● <b>HTTP&amp;HTTPS</b>: Backend requests will be forwarded to the specified backend service.<ul style="list-style-type: none"><li>- <b>Load Balance Channel</b>: Determine whether to use a load balance channel to access the backend service. If you want to select <b>Configure</b>, <a href="#">create a load balance channel</a> in advance.</li><li>- <b>Backend URL</b>: address of the backend service to forward requests to.</li><li>- <b>Timeout (ms)</b>: backend request timeout. The default value is 5000 ms.</li></ul></li><li>● <b>FunctionGraph</b>: Backend requests will be forwarded to a specified function.<ul style="list-style-type: none"><li>- <b>Function URN</b>: unique identifier of a function request. Click <b>Select</b> to select a function.</li><li>- <b>Function Name</b>: automatically displayed after you select a function.</li><li>- <b>Version/Alias</b>: Select the version or alias of the function to be used.</li><li>- <b>Invocation Mode</b>: the mode in which the function is invoked.<ul style="list-style-type: none"><li>- <b>Synchronous</b>: When receiving an invocation request, FunctionGraph immediately processes the request and returns a result. The client closes the connection once it has received a response from the backend.</li><li>- <b>Asynchronous</b>: After receiving an invocation request, FunctionGraph queues the request and returns the result after the request is successfully processed. The server processes the queuing requests one by one when it is idle. The client does not care about the invocation result.</li></ul></li><li>- <b>Timeout (ms)</b>: backend request timeout. The default value is 5000 ms.</li></ul></li></ul> |

| Parameter                    | Description  |
|------------------------------|--|
|                              | <ul style="list-style-type: none"> <li>● <b>Passthrough:</b> Backend requests will be forwarded to the original API backend. To add header parameters to backend requests, click <b>Add Parameter</b>.</li> </ul>  |
| Downgrade Parameter Settings | <p>Specify whether to enable downgrade parameter configuration. When enabled, custom rules take precedence over the default ones configured above:</p> <ul style="list-style-type: none"> <li>● If a custom rule is matched, its trigger conditions and downgrade policy take effect. If the custom rule contains no trigger condition or downgrade policy, the default settings in <b>Trigger Configuration</b> and <b>Backend Downgrade Policy</b> take effect.</li> <li>● If the custom rules are not matched, the default settings take effect.</li> </ul>   |
| Define Parameters            | <p>Define the parameters for rule matching. Click <b>Add Parameter</b> to add rule parameters.</p> <ul style="list-style-type: none"> <li>● <b>Parameter Location:</b> position of a parameter in an API request. <ul style="list-style-type: none"> <li>– <b>path:</b> API request URI. This parameter is configured by default.</li> <li>– <b>method:</b> API request method. This parameter is configured by default.</li> <li>– <b>header:</b> first value of the header parameter in an API request.</li> <li>– <b>query:</b> first value of the query parameter in an API request.</li> <li>– <b>system:</b> a system parameter.</li> </ul> </li> <li>● <b>Parameter:</b> name of a parameter used for rule matching.</li> </ul> |

| Parameter    | Description  |
|--------------|--|
| Define Rules | <p>Customize matching rules for the circuit breaker. Click <b>Add Rule</b> to add rules. The system matches rules from top to bottom.</p> <ul style="list-style-type: none"><li>• <b>Define Rules:</b> Click  to edit the condition expressions. If there are three or more expressions, you can layer them by clicking <b>Set Lower Level</b>.<ul style="list-style-type: none"><li>- =: equal to</li><li>- !=: not equal to</li><li>- <b>pattern:</b> regular expression</li><li>- <b>enum:</b> enumerated values, separated by comma (,)</li></ul></li><li>• For details about how to configure the triggering conditions and backend downgrade, see the instructions for the default settings above.</li></ul> <p>Example: You have enabled <b>Downgrade Parameter Settings</b> and added rules <b>rule01</b> and <b>rule02</b> in sequence. And you have disabled <b>Trigger Configuration</b> and enabled <b>Backend Downgrade Policy</b> for <b>rule01</b>, and have enabled both parameters for <b>rule02</b>. Analysis: The circuit breaker first checks whether the match conditions of <b>rule01</b> are met. If yes, the circuit breaker opens based on the default settings because no trigger condition is defined in <b>rule01</b>, and the backend downgrade policy in <b>rule01</b> is executed. If no, <b>rule02</b> is checked.</p> |

5. Click **OK**.

After a plug-in policy is created, perform [Binding a Plug-in Policy to an API](#) for the policy to take effect for the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab page, filter policies by **Circuit Breaker**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{
  "breaker_condition":{
    "breaker_type":"timeout",
    "breaker_mode":"counter",
    "unhealthy_threshold":30,
    "time_window":15,
```

```
"open_breaker_time":15,
"unhealthy_percentage":51,
"min_call_threshold":20
},
"scope":"share",
"downgrade_default":{
  "type":"http",
  "passthrough_infos":null,
  "func_info":null,
  "mock_info":null,
  "http_info":{
    "isVpc":false,
    "vpc_channel_id":"","",
    "address":"10.10.10.10",
    "scheme":"HTTP",
    "method":"GET",
    "path":"/demo",
    "timeout":5000
  },
  "http_vpc_info":null
},
"downgrade_parameters":[
  {
    "name":"reqPath",
    "type":"path",
    "value":"path",
    "disabled":true,
    "focused":true,
    "id":"92002eqbpilg6g"
  },
  {
    "name":"method",
    "type":"method",
    "value":"method",
    "disabled":true,
    "focused":true,
    "id":"tuvxetsdqvcos8"
  }
],
"downgrade_rules":[
  {
    "rule_name":"rule-test1",
    "parameters":[
      "reqPath",
      "method"
    ],
    "match_regex":"[\\\"reqPath\\\",\\\"=\\\",\\\"/test\\\"]",
    "downgrade_backend":{
      "type":"mock",
      "passthrough_infos":null,
      "func_info":null,
      "mock_info":{
        "status_code":200,
        "result_content":"{status: ok}",
        "headers":[]
      },
      "http_info":null,
      "http_vpc_info":null
    },
    "breaker_condition":{
      "breaker_type":"timeout",
      "breaker_mode":"percentage",
      "unhealthy_threshold":30,
      "time_window":15,
      "open_breaker_time":15,
      "unhealthy_percentage":51,
      "min_call_threshold":20
    }
  }
]
```



## 6.9.6 Configuring A Third-Party Authorizer

ROMA Connect provides flexible extension capabilities for APIs through plug-in policies. Third-party authentication policies call third-party authentication services to authenticate API access. When a user calls an API, ROMA Connect calls the third-party authentication service first and then the backend service if the authentication is successful.

Plug-in policies and APIs are independent of each other. A plug-in policy takes effect for an API only after it is bound to the API. When binding a plug-in policy to an API, you must specify an environment where the API has been published. The plug-in policy takes effect for the API only in the specified environment.

### Constraints

When a user calls an API bound to a third-party authentication policy, the API performs its own authentication and then the third-party authentication.

### Creating a Third-Party Authorizer


1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. In the **Select Policy Type** box, select **Third-Party Authorizer** in the **Plug-ins** area.
4. On the page displayed, configure plug-in policy information.

**Table 6-81** Policy configuration

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a policy name. Using naming rules facilitates future search.   |
| Type                    | The value is fixed as <b>Third-Party Authorizer</b> .  |
| Scope                   | Specify the scope to view the policy. <ul style="list-style-type: none"><li>● <b>Integration application</b>: Each policy belongs to an integration application. Only users who have permission on the integration application can view and use the policy.</li><li>● <b>All</b>: All users in the current instance can view and use the policy.</li></ul> |
| Integration Application | Mandatory when <b>Scope</b> is set to <b>Integration Application</b> .<br>Select an integration application for the policy. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Description             | Describe the policy.   |

| Parameter             | Description   |
|-----------------------|---|
|                       | <b>Policy Content:</b> Configure the policy in a form or script. For details about how to configure a script, see <a href="#">Script Configuration Example</a> .  |
| Load Balance Channel  | Whether to access the third-party authentication service via a load balance channel. If you want to select <b>Configure</b> , <a href="#">create a load balance channel</a> in advance.   |
| Backend URL           | Configure the URL of the third-party authentication service. <ul style="list-style-type: none"><li>• <b>Method:</b> Select the request method of the third-party authentication service.</li><li>• <b>Protocol:</b> Select the request protocol of the third-party authentication service. <b>HTTPS</b> is recommended for transmitting important or sensitive data.</li><li>• <b>Backend Address:</b> Mandatory when <b>Load Balance Channel</b> is set to <b>Skip</b>. Enter the access address of the third-party authentication service in the format of <i>Host.Port</i>. <i>Host</i> indicates the IP address or domain name for accessing the third-party authentication service. If no port is specified, port 80 is for HTTP and port 443 is for HTTPS by default.</li><li>• <b>Load Balance Channel:</b> Mandatory when <b>Load Balance Channel</b> is set to <b>Configure</b>. Select a load balance channel for accessing the third-party authentication service.</li><li>• <b>Path:</b> Enter the request path of the third-party authentication service, for example, <i>/auth</i>.</li></ul> |
| Timeout (ms)          | Enter the timeout interval of a third-party authentication service request. The default value is <b>5000</b> .  |
| Host Header           | Available when <b>Load Balance Channel</b> is set to <b>Configure</b> .<br>Define the host header field carried in the third-party authentication service request.  |
| Brute Force Threshold | IP addresses whose number of attempts to access APIs by third-party authentication within 5 minutes exceeds this threshold will be blocked. They will be unblocked after the current statistical period.<br>For example, if an IP address has failed the third-party authentication more than the configured threshold in the third minute of a statistical period, the address is blocked, and will be unblocked after 2 minutes.  |
| Identity Sources      | Add the header and query parameters in the original API request used for third-party authentication. If not specified, all header and query parameters in the original API request will be used.  |

| Parameter                   | Description  |
|-----------------------------|--|
| Relaxed Mode                | When this option is enabled, ROMA Connect accepts client requests even if the third-party authentication service cannot be connected or returns an error code starting with "5".   |
| Allow Original Request Body | When this option is enabled, the original API request body is used to call the third-party authentication service.   |
| Request Body Size (bytes)   | Available only when <b>Allow Original Request Body</b> is enabled.<br>Size of the request body for calling the third-party authentication service. It cannot exceed the maximum size allowed by the instance. Check out the value of <b>request_body_size</b> in <a href="#">configuration parameters</a> .              |
| Allow Original Request Path | When this option is enabled, the original API request path is added to the end of the third-party authentication request path before the third-party authentication service is invoked.  |
| Return Response             | When this option is enabled, the authentication response is returned on failure.   |
| Allowed Response Headers    | Headers to obtain from the authentication response and send to the backend service with the original request, when the authentication is successful.   |
| Simple Authentication       | When this option is enabled, status codes starting with "2" indicate successful authentication.  |
| Authentication Result       | Available only when <b>Simple Authentication</b> is disabled.<br>Verifies the response header returned by the third-party authentication service. If the response header contains the configured parameter and the corresponding values are the same, the authentication is successful.                                  |
| Blacklist/Whitelist         | When this option is enabled, if the original API request matches the blacklist rule, third-party authentication is performed. If the original API request matches the whitelist rule, third-party authentication is not performed.   |
| Type                        | Mandatory when <b>Blacklist/Whitelist</b> is enabled. <ul style="list-style-type: none"><li>• <b>Whitelist</b>: API requests matching the whitelist rules do not require third-party authentication.</li><li>• <b>Blacklist</b>: API requests matching the blacklist rules require third-party authentication.</li></ul> |

| Parameter  | Description   |
|------------|---|
| Parameters | <p>Available only when <b>Blacklist/Whitelist</b> is enabled.</p> <p>Define the parameters for rule matching. Click <b>Add Parameter</b> to add rule parameters.</p> <ul style="list-style-type: none"><li>● <b>Parameter Location</b>: position of a parameter in an API request.<ul style="list-style-type: none"><li>– <b>path</b>: API request URI. This parameter is configured by default.</li><li>– <b>method</b>: API request method. This parameter is configured by default.</li><li>– <b>header</b>: first value of the header parameter in an API request.</li><li>– <b>query</b>: first value of the query parameter in an API request.</li><li>– <b>system</b>: a system parameter.</li></ul></li><li>● <b>Parameter</b>: name of a parameter used for rule matching.</li></ul> |
| Rules      | <p>Available only when <b>Blacklist/Whitelist</b> is enabled.</p> <p>Customize matching rules for the blacklist or whitelist. Click <b>Add Rule</b> to add rules. The system matches rules from top to bottom.</p> <p><b>Rules</b>: Click  to edit the condition expressions. If there are three or more expressions, you can layer them by clicking <b>Set Lower Level</b>.</p> <ul style="list-style-type: none"><li>● <b>=</b>: equal to</li><li>● <b>!=</b>: not equal to</li><li>● <b>pattern</b>: regular expression</li><li>● <b>enum</b>: enumerated values, separated by comma (,)</li></ul>  |

5. Click **OK**.

After a plug-in policy is created, [bind a plug-in policy to an API](#) for the policy to take effect on the API.

## Binding a Plug-in Policy to an API

1. On the **Policies** tab, filter policies by **Third-Party Authorizer**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the policy to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Script Configuration Example

```
{
  "auth_request": {
    "method": "GET",
    "protocol": "HTTPS",
    "url_domain": "10.10.10.10",
    "timeout": 5000,
    "path": "/auth",
    "vpc_channel_enabled": false,
    "vpc_channel_info": null
  },
  "custom_forbid_limit": 100,
  "carry_body": {
    "enabled": true,
    "max_body_size": 1000
  },
  "auth_downgrade_enabled": true,
  "carry_path_enabled": true,
  "return_resp_body_enabled": false,
  "carry_resp_headers": [],
  "simple_auth_mode_enabled": true,
  "match_auth": null,
  "rule_enabled": false,
  "rule_type": "allow"
}
```

## 6.10 Configuring a Custom Authorizer

### 6.10.1 Creating a Frontend Custom Authorizer

To use your own API calling authentication system, customize a frontend or backend authorizer.

- Frontend custom authorizer: ROMA Connect uses a custom authentication function to authenticate received API requests.
- Backend custom authorizer: The backend service of an API uses a custom authentication function to authenticate backend service requests forwarded by ROMA Connect.

This section describes how to create a frontend custom authorizer. To do this, create a function backend as the authentication function, and use the function backend as the authentication backend in custom authentication.

### Creating a Function Backend for Frontend Authentication

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, click **Create Backend**.
3. On the **Create Backend** page, set backend parameters and click **Create**.
  - **Backend Request Method** must be set to **POST**.
  - The input parameters are not required. The function backend will obtain the values of the header and query parameters from API requests.
  - For details about the settings of other parameters, see [Creating a Function Backend](#).

After the backend is created, the online IDE of the backend is automatically displayed. The backend type defaults to data backend.

4. Develop a function backend.

In the upper left corner of the online IDE, choose **File > Create Function Backend > Blank Template**. In the dialog box displayed, click **Yes**. Compile a function script for security authentication and click **Save**.

The function script used for frontend custom authentication must meet the following conditions:

- For the request parameters obtained using the function script:
  - Header parameter: indicates the identity source parameter in Header defined in custom authentication. The parameter value is transferred from the API request that uses the frontend custom authentication. The called parameter in the function script is in the format of: **body["headers"]["Parameter name"]**.
  - Query parameter: indicates the identity source parameter in Query defined in custom authentication. The parameter value is transferred from the API request that uses the frontend custom authentication. The called parameter in the function script is in the format of: **body["queryStringParameters"]["Parameter name"]**.
  - Body parameter: user data defined when a custom authorizer is created. The format of calling the body parameter is **body["user\_data"]**.

 NOTE

During frontend custom authentication, the header and query parameters of API requests are placed in the **headers** and **queryStringParameters** parameters of the backend request body and transferred to the authentication function. Therefore, when the header and query parameters need to be called in a function script, the parameters need to be obtained from the backend request body. For details about the examples of **headers** and **queryStringParameters** in the backend request body, see [Test Procedure Examples](#).

- Response

The response body cannot be greater than 1 MB. The response content must be in the following format:

```
{
  "status": "allow/deny",
  "context": {
    "user": "abc"
  }
}
```

- **status**: identifies the authentication result. This field is mandatory. Only **allow** or **deny** is supported. **allow** indicates that the authentication is successful, and **deny** indicates that the authentication fails.
- **context**: indicates the authentication response result. This field is mandatory. Only key-value pairs of the string type are supported. The key value does not support JSON objects or arrays. The data in the context is user-defined. After the authentication is successful, the data can be used as a **system parameter** (frontend authentication parameter) and mapped to the backend request

parameter of the API. The system parameter name set in the API backend service must be the same as the parameter name in the context. The parameter name is case sensitive. The parameter name in **context** must start with a letter and contain 1 to 32 characters, including letters, digits, underscores (\_), and hyphens (-).

The following is an example of the Header parameter definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["headers"]["test"]=="abc"){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

The following is an example of the Query parameter definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["queryStringParameters"]["test"]=="abc"){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

The following is an example of the Body parameter definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["user_data"]=="abc"){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

5. Test the function backend.

In the upper right corner, click **Test**. In the **Test Parameters** area, add request parameters required for authentication as defined for the function backend, and click **Test** to send the request. If the value of status in the test result is **allow**, the test is successful.

In the **Test Parameters** area, set backend request parameters. The Header, Query, and Body authentication parameters must be set in the Body parameter of the backend request. Using the preceding script examples, the following describes how to set the authentication parameters:

- Header parameter

```
{  
  "headers":{  
    "test":"abc"  
  }  
}
```

- Query parameter

```
{  
  "queryStringParameters":{  
    "test":"abc"  
  }  
}
```

- Body parameter

```
{  
  "user_data": "abc"  
}
```

6. Deploy the function backend.

After testing, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the function backend.

## Creating a Frontend Custom Authorizer

Before creating a frontend custom authorizer, ensure that the function backend used for frontend custom authentication has been created. Otherwise, create a function backend first. For details, see [Creating a Function Backend for Frontend Authentication](#).

1. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Custom Authorizers** tab, click **Create Custom Authorizer**.
2. On the page displayed, configure custom authorization and click **OK**.

**Table 6-82** Parameters for creating a frontend custom authorizer

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a custom authorizer name. Using naming rules facilitates future search.   |
| Integration Application | Select an integration application for the custom authorizer.  |
| Type                    | Select <b>Frontend</b> .  |
| Function URN            | Select a function backend (must be currently <b>Deployed</b> ) for frontend custom authentication.                    |
| Max. Cache Age (s)      | Set the time (max. 3600) for caching authentication results.<br><b>0</b> : Authentication results will not be cached. |



| Parameter         | Description   |
|-------------------|---|
| Identity Sources  | Add header or query request parameters used for authentication.<br><br>If the value of <b>Max. Cache Age</b> is not <b>0</b> , a request parameter must be added. When the authentication result is cached, this parameter is used as the cache index of the authentication result. |
| Send Request Body | Specify whether to send the API request body to the authentication function.  |
| User Data         | Enter custom parameters, which are used together with <b>Identity Sources</b> for request authentication.   |

## 6.10.2 Creating a Backend Custom Authorizer

To use your own backend request authentication system, customize a frontend or backend authorizer.

- Frontend custom authorizer: ROMA Connect uses a custom authentication function to authenticate received API requests.
- Backend custom authorizer: The backend service of an API uses a custom authentication function to authenticate backend service requests forwarded by ROMA Connect.

This section describes how to create a backend custom authorizer. To do this, create a function backend as the authentication function, and use the function backend as the authentication backend in custom authentication.

### Creating a Function Backend for Backend Authentication

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab, click **Create Backend**.
3. On the **Create Backend** page, set backend parameters and click **Create**.
  - **Backend Request Method** must be set to **POST**.
  - You do not need to set input parameters. The Header and Query parameters are invalid in the function backend used for backend custom authentication.
  - For details about the settings of other parameters, see [Creating a Function Backend](#).

After the backend is created, the online IDE of the backend is automatically displayed. The backend type defaults to data backend.

4. Develop a function backend.  
In the upper left corner of the online IDE, choose **File > Create Function Backend > Blank Template**. In the dialog box displayed, click **Yes**. Compile a function script for security authentication and click **Save**.

The function script used for backend custom authentication must meet the following conditions:

- Request parameter

Body parameter: user data defined when a custom authorizer is created. The format of calling the body parameter is `body["user_data"]`.

- Response

The response body cannot be greater than 1 MB. The response content must be in the following format:

```
{
  "status": "allow/deny",
  "context": {
    "user": "abc"
  }
}
```

- **status**: identifies the authentication result. This field is mandatory. Only **allow** or **deny** is supported. **allow** indicates that the authentication is successful, and **deny** indicates that the authentication fails.
- **context**: indicates the authentication response result. This field is mandatory. Only key-value pairs of the string type are supported. The key value does not support JSON objects or arrays.

The data in the context is user-defined. After the authentication is successful, the data can be used as a **system parameter** (backend authentication parameter) and mapped to the backend request parameter of the API. The system parameter name set in the API backend service must be the same as the parameter name in the context. The parameter name is case sensitive. The parameter name in **context** must start with a letter and contain 1 to 32 characters, including letters, digits, underscores (\_), and hyphens (-).

The following is an example of the user data definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["user_data"]=='abc'){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }
  }else{
  return{
    "status": "deny"
  }
  }
}
```

5. Test the function backend.

In the upper right corner, click **Test**. In the **Test Parameters** area, add request parameters required for authentication as defined for the function backend, and click **Test** to send the request.

The user data definition script in the preceding step is used as an example. Enter the request content `{"user_data": "abc"}` in the body parameter to authenticate backend service requests.

If the value of status in the test result is **allow**, the test is successful.

6. Deploy the function backend.

After the backend is tested, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the function backend.

## Creating a Backend Custom Authorizer

Before creating a backend custom authorizer, ensure that the function backend used for backend custom authentication has been created. Otherwise, create a function API first. For details, see [Creating a Function Backend for Backend Authentication](#).

1. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Custom Authorizers** tab, click **Create Custom Authorizer**.
2. On the page displayed, configure custom authorization and click **OK**.

**Table 6-83** Parameters for creating a backend custom authorizer

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a custom authorizer name. Using naming rules facilitates future search.   |
| Integration Application | Select an integration application for the custom authorizer.  |
| Type                    | Select <b>Backend</b> .   |
| Function URN            | Select a function backend (must be currently <b>Deployed</b> ) for backend custom authentication.                     |
| Max. Cache Age (s)      | Set the time (max. 3600) for caching authentication results.<br><b>0</b> : Authentication results will not be cached. |
| Send Request Body       | Specify whether to send the request body to the function.   |
| User Data               | Enter the parameters for creating a custom authorizer.  |

## 6.11 Configuring Signature Verification for Backend Services

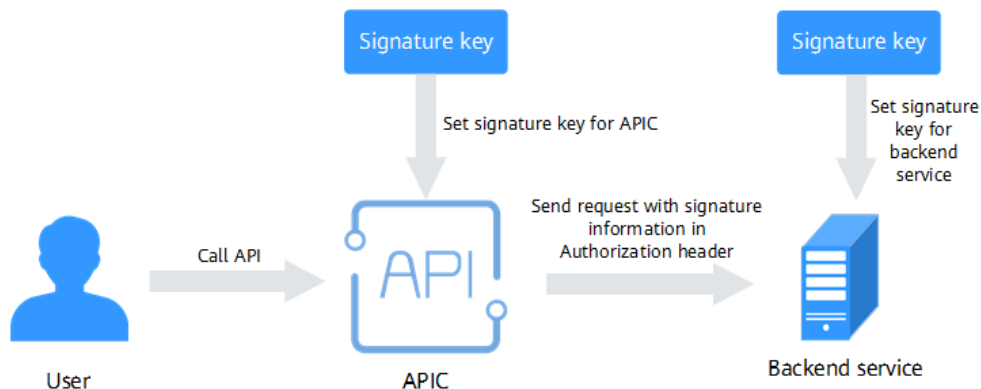
Signature keys are used by backend services to identify ROMA Connect.

A signature key consists of a key and a secret. The signature key takes effect only after it is bound to an API.

### NOTE

An API can be bound to only one signature key in an environment, but a signature key can be bound to multiple APIs.

After a signature key is bound to an API, ROMA Connect uses its key and secret to add signature information to requests sent to the backend service of the API. The backend service needs to sign the requests in the same way. If the signature matches what is included in the **Authorization** header of the requests, the backend service validates the requests sent by ROMA Connect.



## Creating a Signature Key

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Policies**. On the **Policies** tab, click **Create Policy**.
3. On the **Select Policy Type** page, select **Signature Key** in the **Traditional Policy** area.
4. Configure signature key information.

**Table 6-84** Signature key configuration

| Parameter           | Description   |
|---------------------|---|
| Name                | Enter a signature key name. Using naming rules facilitates future search.   |
| Type                | Authentication type.<br>Options: <b>HMAC</b> , <b>AES</b> , <b>Basic auth</b>   |
| Signature Algorithm | AES signature algorithm.<br>Options: <b>aes-128-cfb</b> or <b>aes-256-cfb</b>   |
| Key                 | Set the key based on the signature key type you have selected. <ul style="list-style-type: none"> <li>• <b>Type is HMAC:</b> Enter the key of the key pair for hash-based message authentication code (HMAC) authentication.</li> <li>• <b>Type is Basic auth:</b> Enter the username for authentication.</li> <li>• <b>Type is AES:</b> Enter the key for authentication.</li> </ul> |

| Parameter      | Description   |
|----------------|---|
| Secret         | Set the key based on the signature key type you have selected. <ul style="list-style-type: none"><li>• <b>Type is HMAC:</b> Enter the secret of the key pair for authentication.</li><li>• <b>Type is Basic auth:</b> Enter the password for authentication.</li><li>• <b>Type is AES:</b> Enter the vector for authentication.</li></ul> |
| Confirm Secret | Enter the same secret again.  |

5. Click **OK**.

After the signature key is created, perform [Binding a Signature Key to an API](#) for the signature key to take effect for the API.

## Binding a Signature Key to an API

1. On the **Policies** tab, filter policies by **Signature Key**.
2. Click the name of a policy to go to the details page.
3. On the **APIs** tab, select the environment of the APIs you want to bind the policy to and click **Bind to APIs**.
4. On the page displayed, select the APIs to bind the signature key to.  
APIs can be filtered by API group and API name.
5. Click **OK**.

## Configuring Signature Verification for Backend Services

After binding a signature key to APIs, develop signature verification for backend services to verify request signatures. For details, see [Developing Signature Verification for Backend Services](#).

## 6.12 Configuring API Cascading

API cascading allows you to cascade two ROMA Connect instances in one or more regions. This lets one instance use an API of the other instance as its backend service for cross-instance API calling. A dedicated authentication channel will be used for API cascading to prevent authentication conflict.

- Cascading instance: uses an API of the other instance as its backend service.
- Cascaded instance: provides its API to the other instance as a backend service.


API cascading enables you to provide APIs in one instance for another instance to improve the reuse capability of API assets, without having to deploy backend services in different instances repeatedly.

### Prerequisites

- The two instances can communicate with each other.


- If the two instances are located in different networks and communicate with each other through an air wall, their IP address and port number must be configured on the air wall. In addition, the air wall must use the TCP protocol for secure access. A dedicated VPN or tunnel can also be used for cross-network interworking.

## Procedure

1. Enable cascading for the cascaded instance.
  - a. Log in to the ROMA Connect console that displays the cascaded instance. On the **Instances** page, click **View Console**.
  - b. Choose **Instance Information > Configuration Parameters**, and locate the **cascade** parameter.
  - c. Click **Edit** on the right of the parameter, set **Current Value** to **on**, and click **Save**.
  - d. Click  on the left of the parameter and configure the following parameters.

**Table 6-85** Parameters related to the cascading function

| Parameter            | Description   |
|----------------------|---|
| cascade_auth_key     | Encryption key used for authentication between two APIs.<br>This must be the same for the cascaded and cascading instance pair.   |
| cascade_instance_ids | IDs of the instances that are allowed to cascade the current instance. Use commas (,) to separate multiple (max. 5) instance IDs. |

2. Enable cascading for the cascading instance.
  - a. Log in to the ROMA Connect console that displays the cascading instance. On the **Instances** page, click **View Console**.
  - b. Choose **Instance Information > Configuration Parameters**, and locate the **cascade** function.
  - c. Click **Edit** on the right, set **Current Value** to **on**, and click **Save**.
  - d. Click  on the left and configure the following parameters.

**Table 6-86** Parameters related to the cascading function

| Parameter        | Description   |
|------------------|---|
| cascade_auth_key | Encryption key used for authentication between two APIs.<br>This must be the same for the cascaded and cascading instance pair. |

| Parameter            | Description   |
|----------------------|---|
| cascade_instance_ids | This parameter is not required for cascading instances. |

3. Create a load balance channel from the cascading instance to the cascaded instance.
  - a. In the navigation pane of the cascading instance console, choose **API Connect > API Policies**. On the **Load Balance Channels** tab, click **Create Load Balance Channel**.
  - b. On the page displayed, configure load balance channel information.

**Table 6-87** Basic information

| Parameter         | Description   |
|-------------------|---|
| Name              | Enter a load balance channel name.<br>Using naming rules facilitates future search.   |
| Port              | Enter the access port number of the ECS in the load balance channel.<br>The port number is the protocol used by the API in the cascaded instance.<br><b>80</b> : HTTP<br><b>443</b> : HTTPS |
| Routing Algorithm | Select an algorithm for routing backend service requests. The load balance channel determines which server the algorithm will send the requests to.   |

- c. Configure servers. To access APIs of the cascaded instance, select **Specify IP addresses**.
- d. Click **Create Server Group**. In the dialog box displayed, configure group information and click **OK**.  
Servers can be added to different groups.

**Table 6-88** Server group configuration

| Parameter   | Description  |
|-------------|--|
| Group Name  | Enter a server group name. Using naming rules facilitates future search.   |
| Weight      | Enter the weight of the server group. The larger the weight, the more requests can be forwarded to the servers in the group. |
| Description | Enter a brief description of the server group.   |

- e. Click **Add Backend Server Addresses** and configure backend servers.

**Table 6-89** Backend server information

| Parameter              | Description   |
|------------------------|---|
| Backend Server Address | Enter the API access address of the cascaded instance. <ul style="list-style-type: none"><li>IP address format:<ul style="list-style-type: none"><li>Set this parameter to the EIP of the cascaded instance if the two instances communicate with each other over a public network.</li><li>Set this parameter to the APIC connection address of the cascaded instance if the two instances communicate with each other over a VPC intranet.</li></ul></li><li>Domain name format: Enter the access domain name of the API.</li></ul> |
| Standby Node           | Enabled: The backend server serves as a standby node. It works only when all non-standby nodes are faulty.  |
| Port                   | Enter the access port number of the backend server.<br><b>0</b> : uses the load balance channel port.   |
| Server Status          | Specify whether to enable the server. Requests are distributed to the server only after it is enabled.  |

- f. Configure the health check.

The health check function is enabled by default. If you do not need the health check, disable this function.

**Table 6-90** Health check parameters

| Parameter              | Description   |
|------------------------|---|
| Protocol               | Select the protocol used for the health check.<br>Options: <b>TCP, HTTP, HTTPS</b>  |
| Two-way Authentication | Available for <b>Protocol</b> set to <b>HTTPS</b> .<br>Specify whether to enable two-way authentication between ROMA Connect and backend servers. |
| Path                   | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Enter the health check URL.   |



| Parameter           | Description  |
|---------------------|--|
| Method              | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Select the HTTP request method used for the health check.<br>Options: <b>GET</b> or <b>HEAD</b>  |
| Health Check Port   | Enter the destination port of the health check.<br>Default: port number configured for the load balance channel  |
| Healthy Threshold   | Number of consecutive successful checks required for an ECS to be considered healthy.<br>Example: If set to <b>2</b> , ROMA Connect declares the ECS status to be healthy when the check is successful twice in a row. |
| Unhealthy Threshold | Number of consecutive failed checks required for an ECS to be considered unhealthy.<br>Example: If set to <b>5</b> , ROMA Connect declares the ECS status to be abnormal when the check fails five times in a row.     |
| Timeout (s)         | Response timeout of a health check, in seconds. If no response is received within this time, the health check fails.   |
| Interval (s)        | Interval between consecutive checks.   |
| Response Codes      | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>When the server returns a specified HTTP response code, the server declares the response to be successful.<br>Supports multiple response codes.  |

- g. Click **Finish**.
4. On the cascading instance, create an API and set the backend address to the API in the cascaded instance.

For details about how to create an API, see [Creating an API](#). Only the configuration of defining the backend service is different between the cascading and the cascaded instances, as shown as follows.

**Table 6-91** Backend service access parameters

| Parameter    | Description   |
|--------------|---|
| Backend Type | Select a backend service type. When the API of the cascaded instance is used as the backend service, select <b>HTTP/HTTPS</b> . |

| Parameter            | Description   |
|----------------------|---|
| Load Balance Channel | Determine whether to use a load balance channel to access backend services. When the API of the cascaded instance is used as the backend service, select <b>Configure</b> .   |
| URL                  | <p>Configure the URL of the backend service.</p> <ul style="list-style-type: none"><li>• <b>Method</b>: Select the request method of the backend service based on the API request method of the cascaded instance.</li><li>• <b>Protocol</b>: Select the request protocol used by the backend service based on the API request protocol of the cascaded instance.</li><li>• <b>Load Balance Channel</b>: Select the load balance channel created in <a href="#">3</a>.</li><li>• <b>Path</b>: Enter the request path of the backend service, for example, <code>/getUserInfo/{userId}</code>. A URL can have multiple path parameters, each enclosed by braces. If the path needs to contain an environment variable, enclose the environment variable in number signs (#)<br/>Example: <code>/#path#</code><br/>Environment variable names are case sensitive.<br/>Multiple environment variables can be added.<br/>Example: <code>/#path##request#</code></li></ul> |
| Cascading Flag       | Determine whether to use the cascading mode to access backend services. Enable this option.   |
| Host Header          | Define the Host header field carried in the backend service request. If you have specified <b>Backend Server Address</b> with an IP address when creating a load balance channel in <a href="#">3</a> , set <b>Host Header</b> to the domain name of the API of the cascaded instance.  |
| Timeout (ms)         | Enter the timeout interval of a backend service request.<br>Default: <b>5000</b>  |
| Retries              | <p>Number of retries after ROMA Connect fails to call the backend service.</p> <ul style="list-style-type: none"><li>• <b>-1</b>: The retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>• <b>0 to 10</b>: This parameter is enabled and will make the configured number of retries. The number must be less than that of backend servers enabled in the load balance channel.</li></ul>  |

| Parameter              | Description  |
|------------------------|--|
| Two-Way Authentication | Available for <b>Protocol</b> set to <b>HTTPS</b> .<br>Determine whether to enable two-way authentication between ROMA Connect and backend services. When the API of the cascaded instance is used as the backend service, do not enable two-way authentication. |
| Backend Authentication | Determine whether to enable backend authentication. When the API of the cascaded instance is used as the backend service, do not enable backend authentication.  |

# 7 Service Integration Guide (Old Edition)

---

[Usage Introduction](#)

[Exposing APIs](#)

[Creating and Exposing Data APIs](#)

[Creating and Exposing Function APIs](#)

[Calling an API](#)

[Managing APIs](#)

[Managing Custom Backends](#)

[Managing Control Policies](#)

[Managing Plug-ins](#)

[Configuring a Custom Authorizer](#)

[Configuring Signature Verification for Backend Services](#)

[Configuring API Cascading](#)

## 7.1 Usage Introduction

### Function Description

APIC is the API integration component of ROMA Connect. It encapsulates APIs, data sources, and custom functions into standard RESTful APIs, and then exposes them to external systems. ROMA Connect has the following advantages for service integration:

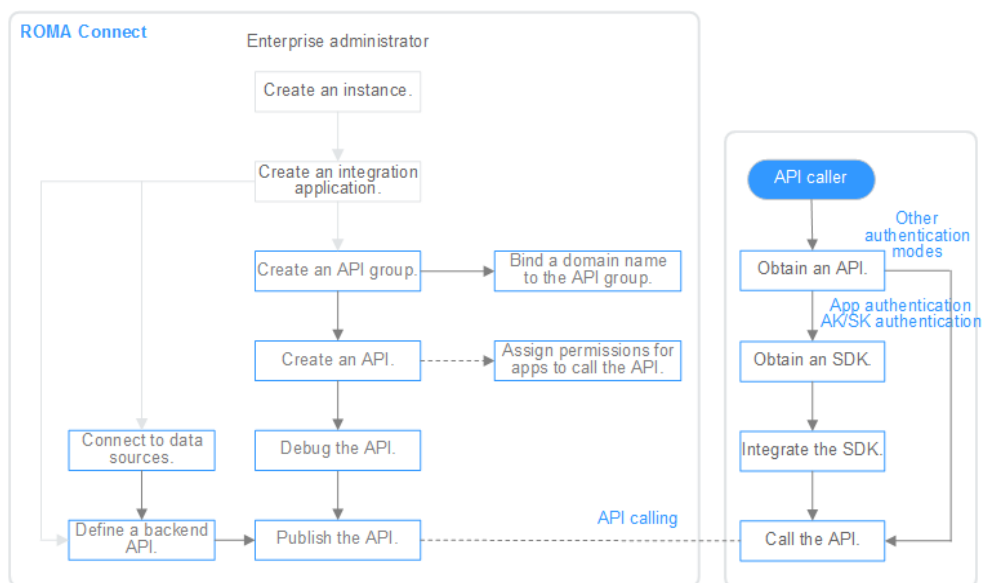
- **Convenient API management**  
ROMA Connect provides full-lifecycle management for APIs, including creating, debugging, publishing, taking offline, authorizing, editing, and deleting APIs.
- **Custom API backend services**  
ROMA Connect provides two types of backends.

- Data backends, where data sources are exposed as APIs. For details about the supported source types, see [Data Sources Supported by APIC](#).
- Function backends: Function capabilities are exposed as APIs.
- **API monitoring portal**  
ROMA Connect provides a visualized dashboard for API calling and analysis of the performance metrics related to API calling and identifies potential risks that may affect services.
- **Multi-layer security protection**  
ROMA Connect provides multiple authentication modes, refined request throttling, and strict access control for secure API calling.

## Process Flow

The following figure shows how ROMA Connect integrates services.

**Figure 7-1** Process of using ROMA Connect for service integration



1. You have **created an instance and integration application**.

2. **Expose an API.**

- Exposing APIs

- i. **Create an API group.**

Each API belongs to an API group, so create one before creating an API.

- ii. **Bind a domain name to the API group.**

Before exposing an API, bind an independent domain name to the API group so that users can access the API.

(Optional) Before binding an independent domain name to an API, use the default subdomain name to test API calling. ROMA Connect limits the number of times that this default subdomain name can be accessed (1000 times a day).

- iii. **Create an API.**  
Encapsulate existing backend services into standard RESTful APIs and expose them to external systems.
  - iv. **Debug the API.**  
Verify that the API service functions are normal using the ROMA Connect debugging function.
  - v. **Publish the API.**  
Publish an API in an environment so that it can be called.
  - vi. (Optional) **Grant permissions for APIs.**  
For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.
- Creating and exposing a data API
    - i. **Connect to data sources.**  
Connect to a data source to ensure that data can be read from the data source.
    - ii. **Create a data API.**  
Define data sources as APIs and expose them to external systems through ROMA Connect.
    - iii. (Optional) **Grant permissions for APIs.**  
For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.
  - Creating and exposing a function API
    - i. **Create a function API.**  
Define custom functions as APIs and expose them to external systems.
    - ii. (Optional) **Grant permissions for APIs.**  
For APIs that use App authentication: After an API is authorized to a specified integration application, the key and secret of the authorized integration application authenticate API requests as a security measure.
3. **Call the API.**  
Obtain the API and its access address to call the API. This step requires different authentication operations depending on the authentication mode.

## 7.2 Exposing APIs

## 7.2.1 Creating an API Group

### Overview

An API group is a set of APIs the same type of services use. An API developer creates an API group to manage all APIs in the group. Each API belongs to an API group. Before creating an API, create an API group.

### Prerequisites

Each API group must belong to an integration application. Before creating an API group, ensure that an integration application is available, or [create one](#).

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **API Groups** tab page, click **Create**.
3. In the **Create API Group** dialog box, enter the group-related parameters and click **OK**.

**Table 7-1** API group parameters

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter an API group name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Type                    | Type of the API group. <ul style="list-style-type: none"><li>• <b>Integration application:</b> An API group belongs to a specific integration application. Only users who have permissions on the integration application can view and perform operations on the API group.</li><li>• <b>Global:</b> All users can view and perform operations on the API group.</li></ul> |
| Integration Application | This parameter is mandatory only if <b>Integration application</b> is selected for <b>Type</b> .<br>Select the API group's integration application. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Description             | Enter a brief description of the API group.  |

#### NOTE

- The system allocates a subdomain name to the API group for internal testing. The subdomain name can be accessed up to 1000 times a day.
- When opening APIs, you must bind an independent domain name to the API group.

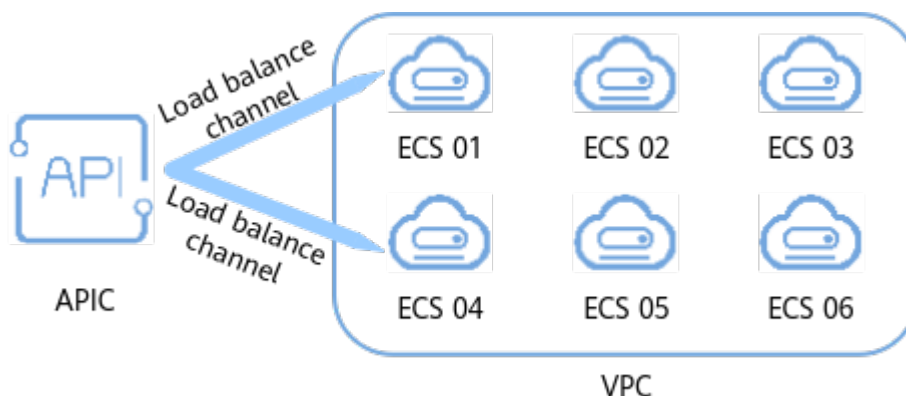
## 7.2.2 (Optional) Creating a Load Balance Channel

### Overview

This channel allows ROMA Connect to access backend services deployed on servers in load balancing mode (direct access to ECSs in the same VPC, or to ECSs in other VPCs and private servers by specifying IP addresses).

For example, six ECSs deployed in a VPC have a load balance channel to reach ECS 1 and ECS 4. ROMA Connect can access these two ECSs through the channel.

**Figure 7-2** Accessing the ECSs in a load balance channel



### Prerequisites

- The network between ROMA Connect and the servers in the load balance channel is normal.
- You have the **VPC Administrator** permission.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Load Balance Channels** tab page, click **Create**.
3. On the **Create Load Balance Channel** page, configure load balance channel information.
  - a. Configure basic information about the load balance channel.

**Table 7-2** Parameters for configuring the load balance channel

| Parameter | Description  |
|-----------|--|
| Name      | Enter a load balance channel name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Port      | Enter the access port number in the load balance channel.  |



| Parameter              | Description   |
|------------------------|---|
| Distribution Algorithm | Select a distribution algorithm for backend service requests. The load balance channel determines the server to which the request is to be sent based on the distribution algorithm.  |
| Backend Server Type    | Select the member type in the load balance channel. The member type is a one-time configuration and cannot be changed after you create the VPC channel. <ul style="list-style-type: none"><li>• <b>Cloud server:</b> Add a load balance channel member by selecting a cloud server.</li><li>• <b>IP address:</b> Add a load balance channel member by entering a server IP address.</li></ul> |

- b. Create a server group.
  - i. Click **Create Server Group**.
  - ii. In the **Create Server Group** dialog box, configure the group information and click **OK**.

You can divide servers into multiple groups.

**Table 7-3** Server group configuration

| Parameter   | Description  |
|-------------|--|
| Group Name  | Name of a server group. Set this parameter as planned. It is recommended that you enter a name based on naming rules to facilitate search. |
| Weight      | Enter the weight of the server group. A larger weight indicates that more requests are forwarded to the server in the group.               |
| Description | Enter a brief description of the group.  |

- c. Add cloud servers or server addresses to the load balance channel based on the backend server type configured.
  - Add a cloud server.
    - 1) Click **Select ECS**.
    - 2) In the **Select ECS** dialog box, select a subnet, select the ECS to be added, and click **OK**.
  - Add a backend server address.
    - 1) Click **Add Backend Server Address**.
    - 2) On the page displayed, configure backend server information.

**Table 7-4** Backend server information

| Parameter              | Description   |
|------------------------|---|
| Backend Server Address | Enter the IP address of the backend server.   |
| Standby Node           | After you enable this option, the backend server serves as a standby node. It works only when all non-standby nodes are faulty. |
| Port                   | Enter the access port number of the backend server. If the port number is 0, the port of the load balance channel is used.      |
| Server Status          | Specify whether to enable the server. Requests are distributed to the server only after it is enabled.                          |

## d. Configure the health check.

The health check function is enabled by default. If you do not need to perform the health check, disable this parameter.

**Table 7-5** Health check configurations

| Parameter              | Description  |
|------------------------|--|
| Protocol               | Select the protocol used for the health check. The value can be <b>TCP</b> , <b>HTTP</b> , or <b>HTTPS</b> .   |
| Two-Way Authentication | This parameter is available only if <b>Protocol</b> is set to <b>HTTPS</b> .<br>Specify whether to enable two-way authentication between ROMA Connect and backend servers.   |
| Path                   | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Enter the health check URL.  |
| Method                 | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Select the HTTP request method used for the health check. The value can be <b>GET</b> or <b>HEAD</b> .   |
| Health Check Port      | Enter the destination port of the health check. By default, the port number configured for the load balance channel is used.   |
| Healthy Threshold      | Number of consecutive successful checks required for a server to be considered healthy. For example, if <b>Healthy Threshold</b> is set to <b>2</b> , the server is considered healthy when it passes two consecutive health checks. |

| Parameter           | Description   |
|---------------------|---|
| Unhealthy Threshold | Number of consecutive failed checks required for a server to be considered unhealthy. For example, if <b>Unhealthy Threshold</b> is set to <b>5</b> , the server is considered unhealthy when it fails five consecutive health checks.    |
| Timeout Interval    | Response timeout of a health check, in seconds. If no response is received within the specified duration, the health check fails.   |
| Interval (s)        | Interval between two consecutive checks, in seconds.  |
| Response Codes      | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>When the server returns a specified HTTP response code, the server considers the response as successful. Multiple response codes can be specified at the same time. |

4. Click **Finish**.

## 7.2.3 Creating an API

### Overview

Create APIs to encapsulate existing backend services into standard RESTful APIs and make them accessible to other users.

### Prerequisites

- Each API must belong to an integration application. Before creating an API, ensure that an integration application is available. Otherwise, [create an integration application](#) first.
- Each API must belong to an API group. Before creating an API, ensure that the API group is available. Otherwise, [create an API group](#) first.
- If you need to use a load balance channel to access the server where the backend service is located, [create a load balance channel](#) first.
- If you need to use custom API authentication, [create a custom authorizer](#) first.
- Before creating an API, ensure that the ROMA Connect instance can communicate with the network where your backend service resides.
  - Same VPC: The instance can directly access the backend service.
  - Two VPCs in the same region: Connect the instance and the backend service using a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.

- To enable cross-VPC private access, **configure the routes** between the instance and the subnet where the backend service resides.
- To use FunctionGraph as an API backend service, the user must be assigned the **FunctionGraph Administrator** role.

## Setting Basic Information

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, click **Create API**.
3. On the **Create API** page, configure basic information about the API.

**Table 7-6** Basic API information

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter an API name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| API Group               | Select the API group to which the API belongs. If no API group is available, click <b>Create API Group</b> on the right to create an API group.<br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API.   |
| Integration Application | This parameter is available only if the type the API group is set to <b>Global</b> .<br>Select the integration application to which the API belongs. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Visibility              | Determine whether to make the API available on the marketplace. <ul style="list-style-type: none"><li>• <b>Public</b>: The API can be made available on the marketplace.</li><li>• <b>Private</b>: The API will not be made available on the marketplace even if the API group to which the API belongs is made available on the marketplace.</li></ul> |

| Parameter                 | Description   |
|---------------------------|---|
| Security Authentication   | <p>Select the authentication mode of the API. The App authentication mode is recommended.</p> <ul style="list-style-type: none"><li>• <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an authorized integration application. APIs using this mode can be called by all users.</li><li>• <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK. APIs using this mode can be called only by users on the same cloud service platform.</li><li>• <b>Custom</b>: The custom function API is used for authenticating API requests. APIs using this mode can be called by all users.</li><li>• <b>None</b>: Authentication is not required for API requests. APIs using this mode can be called by all users.</li></ul> |
| Simple Authentication     | <p>This parameter is available only if <b>Security Authentication</b> is set to <b>App</b>.</p> <p>This parameter specifies whether to use simple security authentication for API calling. It takes effect only when the API request protocol is HTTPS. Once enabled, AppCodes are included when an API is called, and signature verification is not required for API requests.</p>   |
| Two-Factor Authentication | <p>This parameter is available only if <b>Security Authentication</b> is set to <b>App</b> or <b>IAM</b>.</p> <p>This parameter specifies whether to perform two-way authentication on API calling. If this parameter is selected, the custom function API is also used to authenticate API requests when <b>APP</b> or <b>IAM</b> has been selected for <b>Security Authentication</b>.</p>  |
| Custom Authorizer         | <p>This parameter is mandatory if <b>Security Authentication</b> is set to <b>App</b> or <b>IAM</b> and <b>Two-Factor Authentication</b> is enabled, or if <b>Security Authentication</b> is set to <b>Custom</b>.</p> <p>Select a frontend custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a frontend custom authorizer.</p>  |
| Tag Name                  | Add tags for the API to quickly filter and search for the API.  |
| Description               | Enter a brief description of the API.   |

4. Click **Next**.

## Defining API Request

1. Define request information of an API.

**Table 7-7** API request configuration

| Parameter        | Description  |
|------------------|--|
| Request Protocol | Select the request protocol used by the API. The value can be <b>HTTP</b> , <b>HTTPS</b> , or <b>HTTP&amp;HTTPS</b> . <b>HTTPS</b> is recommended for transmitting important or sensitive data.  |
| Path             | Enter the request path of the API, in the <i>/getUserInfo/{userId}</i> format. A URL can have multiple path parameters, each enclosed by braces. <ul style="list-style-type: none"><li>• Each path parameter must match the entire segment between slashes (/). For example, <i>/abc{userId}</i> is not supported. If <b>Matching</b> is set to <b>Exact match</b>, a plus sign (+) can be added to the end of the path parameter, for example, <i>/users/{p+}</i>. The variable <i>p</i> matches the segments between one or multiple pairs of slashes (/).</li><li>• <b>Path</b> parameters contained in the request path must be defined as request parameters.</li><li>• The value of <b>Path</b> is case sensitive.</li></ul>   |
| Matching         | Select the matching mode of the API request path. <ul style="list-style-type: none"><li>• <b>Exact match</b>: The path in an API request must be the same as the value of <b>Path</b>.</li><li>• <b>Prefix match</b>: The path in an API request must be prefixed with the value of <b>Path</b>. For example, if <b>Path</b> is set to <i>/test/AA</i> and <b>Matching</b> is set to <b>Prefix match</b>, the API can be accessed using <i>/test/AA/BB</i> or <i>/test/AA/CC</i> but cannot be accessed using <i>/test/AACC</i>.</li></ul> <p><b>NOTE</b><br/>When <b>Matching</b> is set to <b>Prefix match</b>, the rest characters of the API access address except the matched prefix are transparently transmitted to the backend service. For example, if the request path is set to <i>/test</i> and the backend request path is set to <i>/test2</i>, the request path received by the backend service is <i>/test2/AA/CC</i> when <i>/test/AA/CC</i> is used to access the API.</p> |
| Method           | Select the request method of the API. <b>ANY</b> indicates that the API can be accessed using any request method.  |
| CORS             | This parameter specifies whether CORS is supported for the API.<br><br>For security purposes, the browser restricts the cross-domain requests from being initiated from a page script. In this case, the page can access only the resources from the same source. CORS allows the browser to send XMLHttpRequest requests to the server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a> .   |

| Parameter                         | Description  |
|-----------------------------------|--|
| (Optional)<br>Input<br>Parameters | <p>Define API request parameters based on the site requirements. Parameters contained in the request path must be defined as input parameters.</p> <p>In the <b>Input Parameters</b> area, click <b>Add Input Parameter</b> and add request parameters of the API.</p> <ul style="list-style-type: none"> <li>• <b>Name:</b> name of a request parameter. If <b>Location</b> is set to <b>PATH</b>, the parameter name must be the same as that in the request path.</li> <li>• <b>Location:</b> position of the request parameter in the API request. The value can be <b>PATH</b>, <b>HEADER</b>, or <b>QUERY</b>.</li> <li>• <b>Type:</b> data type of the request parameter. The value can be <b>STRING</b> or <b>NUMBER</b>.</li> <li>• <b>Mandatory:</b> specifies whether a request parameter is mandatory in an API request.</li> <li>• <b>Passthrough:</b> indicates whether to transparently transmit request parameters to the backend service.</li> <li>• <b>Default Value:</b> The default value of the request parameter can be set only when <b>Mandatory</b> is set to <b>No</b>.</li> <li>• <b>Enumerated Value:</b> enumerated value of the request parameter. The value of the request parameter can only be selected from the enumerated values. Use commas (,) to separate multiple enumerated values.</li> <li>• <b>Max. Length/Max. Value:</b> If <b>Type</b> is set to <b>STRING</b>, set this parameter to the maximum character string length as the parameter value. If <b>Type</b> is set to <b>NUMBER</b>, set this parameter to the maximum value as the parameter value.</li> <li>• <b>Min.Length/Min.Value:</b> If <b>Type</b> is set to <b>STRING</b>, set this parameter to the minimum string length as the parameter value. If <b>Type</b> is set to <b>NUMBER</b>, set this parameter to the minimum value as the parameter value.<br/>Setting both <b>Min. Length/Min. Value</b> and <b>Max. Length/Max. Value</b> to <b>0</b> indicates no limit.</li> <li>• <b>Example:</b> example of a request parameter value.</li> <li>• <b>Description:</b> description of a request parameter.</li> </ul> <p><b>NOTE</b></p> <ul style="list-style-type: none"> <li>• The parameter name is not case-sensitive. It cannot be <b>x-stage</b> or start with <b>x-apig-</b> or <b>x-sdk-</b>.</li> <li>• Ensure that the name of a header parameter is not <b>Authorization</b> or <b>X-Auth-Token</b> (not case-sensitive).</li> </ul> |

| Parameter | Description   |
|-----------|---|
| Body      | This parameter is available only if <b>Method</b> is set to <b>POST</b> , <b>PUT</b> , <b>PATCH</b> , or <b>ANY</b> .<br>Enter the description of the request body in the API request to help the API caller understand how to correctly encapsulate the API request. |

2. Click **Next**.

## Defining Backend Requests

1. Configure the basic definition of the default backend.

### NOTE

If FunctionGraph is not deployed in the current environment, the backend type cannot be set to **FunctionGraph**.

**Table 7-8** API backend service parameters

| Backend Type | Parameter                | Description   |
|--------------|--------------------------|---|
| HTTP/HTTPS   | Protocol                 | Select the request protocol used by the backend service. WebSocket communication is supported. <b>HTTPS</b> is recommended for transmitting important or sensitive data.                  |
|              | Method                   | Select the request method of a backend service. <b>ANY</b> indicates that the backend service can be accessed using any request method.   |
|              | Use Load Balance Channel | This parameter specifies whether to use the load balance channel to access backend services. If you want to select <b>Yes</b> , <a href="#">create a load balance channel</a> in advance. |



| Backend Type | Parameter            | Description  |
|--------------|----------------------|--|
|              | Backend Address      | <p>This parameter is mandatory only if <b>Use Load Balance Channel</b> is set to <b>No</b>.</p> <p>Enter the access address of the backend service in the format of <i>Host:Port</i>. <i>Host</i> indicates the IP address or domain name for accessing the backend service. If no port is specified, port 80 is used for HTTP by default, and port 443 is used for HTTPS by default.</p> <ul style="list-style-type: none"> <li>• If the backend address needs to contain environment variables, use <i>#Variable name#</i> to add the environment variables to the backend address, for example, <b>#ipaddress#</b>. Multiple environment variables can be added, for example, <b>#ipaddress##test#</b>.</li> <li>• If you want to call a custom backend by using its backend request address, add two built-in gateway parameters to <b>system parameters</b>. <ul style="list-style-type: none"> <li>- <b>apild</b>: Set <b>Backend Parameter Name</b> to <b>x-auth-app</b> and <b>Backend Parameter Location</b> to <b>HEADER</b>.</li> <li>- <b>providerAppId</b>: Set <b>Backend Parameter Name</b> to <b>x-ld-appid</b> and <b>Backend Parameter Location</b> to <b>HEADER</b>.</li> </ul> </li> </ul> |
|              | Load Balance Channel | <p>This parameter is mandatory only if <b>Use Load Balance Channel</b> is set to <b>Yes</b>.</p> <p>Select the load balance channel used to access the backend service.</p>  |
|              | Cascading Flag       | <p>This parameter is available only when <b>cascade</b> is set to <b>on</b> for <b>instance parameter configuration</b> and <b>Use Load Balance Channel</b> is set to <b>Yes</b>.</p> <p>Whether to allow cascading access to backend services. This parameter must be enabled when <b>API cascading</b> is enabled.</p>   |
|              | Host Header          | <p>This parameter is available only when <b>Use Load Balance Channel</b> is set to <b>Yes</b>.</p> <p>Define the host header field carried in the backend service request.</p>   |

| Backend Type | Parameter              | Description   |
|--------------|------------------------|---|
|              | Path                   | Enter the request path of the backend service, in the <i>/getUserInfo/{userId}</i> format. A URL can have multiple path parameters, each enclosed by braces.<br>If the path needs to contain an environment variable, enclose the environment variable in number signs (#), for example, <i>/#path#</i> . Multiple environment variables can be added, for example, <i>/#path##request#</i> .   |
|              | Timeout (ms)           | Enter the timeout interval of a backend service request. The default value is <b>5000</b> .   |
|              | Retry Times            | Number of retry times after ROMA Connect fails to call the backend service. <ul style="list-style-type: none"> <li>• If the value is <b>-1</b>, the retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li> <li>• If the value ranges from 0 to 10, this parameter is enabled, and retries are performed based on the configured value.</li> </ul> <p><b>NOTE</b><br/>If <b>Use Load Balance Channel</b> is set to <b>Yes</b>, the number of retries must be less than the number of enabled backend servers in the load balance channel.</p> |
|              | Two-Way Authentication | This parameter is available only if <b>Protocol</b> is set to <b>HTTPS</b> .<br>Determine whether to enable two-way authentication between ROMA Connect and backend services. If you enable this option, you also need to configure the certificate for client authentication.  |
|              | Backend Authentication | Determine whether to enable backend authentication. If this option is enabled, a custom function API is used to perform security authentication on backend service requests.  |

| Backend Type  | Parameter              | Description   |
|---------------|------------------------|---|
|               | Custom Authorizer      | This parameter is mandatory only if <b>Backend Authentication</b> is enabled.<br>Select a custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer.   |
| FunctionGraph | Function URN           | Enter the ID of a function request. Click <b>Add</b> to add a function URN as that of a backend service.  |
|               | Version                | Select the version of the function to be used.  |
|               | Invocation Mode        | Select the invocation type of the function. <ul style="list-style-type: none"><li>• <b>Synchronous</b>: When receiving an invocation request, FunctionGraph processes the request immediately and returns a result. The client closes the connection once receiving a response from the backend.</li><li>• <b>Asynchronous</b>: After receiving a calling request, FunctionGraph queues the request and returns the result after the request is successfully executed. The server processes the queuing requests one by one when it is idle. The client does not care about the calling result.</li></ul> |
|               | Timeout (ms)           | Enter the timeout interval of a backend service request. The default value is <b>5000</b> .   |
|               | Backend Authentication | Determine whether to enable backend authentication. If this option is enabled, a custom function API is used to perform security authentication on backend service requests.  |
|               | Custom Authorizer      | This parameter is mandatory only if <b>Backend Authentication</b> is enabled.<br>Select a custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer.   |
|               |                        |   |

| Backend Type   | Parameter              | Description   |
|--|------------------------|---|
| <b>Mock</b><br><b>NOTE</b><br>If the backend service is unavailable, you can use the Mock mode to return the expected result to the API caller for debugging and verification. | Status Code            | Select the HTTP status code returned by the API.  |
|  | Response               | Enter the API response result.<br>For example, if <b>Response</b> is set to <b>Successful Info</b> , the API always returns <b>Successful Info</b> .  |
|  | Backend Authentication | Determine whether to enable backend authentication. If this option is enabled, a custom function API is used to perform security authentication on backend service requests.  |
|  | Custom Authorizer      | This parameter is mandatory only if <b>Backend Authentication</b> is enabled.<br>Select a custom authorizer you have created. If no custom authorizer is available, click <b>Create Custom Authorizer</b> on the right to create a backend custom authorizer. |
|  | Response Header        | Customize the response header information of the API response.<br>Click <b>Add Response Header</b> and enter the parameter name, value, and description.  |

 **NOTE**

- If an environment variable is defined in the backend request path, the API cannot be debugged.
  - For environment variables defined in the backend request path, corresponding variables and their values must be created. Otherwise, the APIs cannot be published because there will be no values that can be assigned to the variables.
  - Environment variable names are case-sensitive.
2. (Optional) Configure the backend parameters for the default backend to map the request parameters transferred when the API is called to the corresponding location in the backend service request. If no request parameter is defined in the API request, skip this step.
    - a. In the **Backend Parameters** area, add backend parameters in either of the following ways:
      - Click **Import Input Parameter** to add all defined API request parameters to the backend parameters.
      - Click **Add Backend Parameter Mapping** to add backend parameters one by one as required.

- b. Modify the mapping between API request parameters and backend parameters.
  - The name and location of the backend parameters can be different from those of the input parameters.
  - If **Backend Parameter Location** is **PATH**, the backend parameter name must be the same as the parameter name in **Path**.
  - The backend parameter name is not case-sensitive. It cannot be **x-stage** or start with **x-apig-** or **x-sdk-**.
  - If **Backend Parameter Location** is **HEADER**, the parameter name is not case-sensitive.

The parameters and backend request path in the following table are used as an example. Parameters **test01** and **test03** are located in **PATH** and **QUERY** in an API request. Through parameter mappings, the backend service receives the values of **test01** and **test03** in **HEADER**. Parameter **test02** is located in **HEADER** in an API request. Through the parameter mapping, the backend service receives the value of **test02** using **test05** in **PATH**.

| Input Parameter Name                    | Input Parameter Location | Backend Parameter Name | Backend Parameter Location |
|---|--------------------------|------------------------|----------------------------|
| test01                                  | PATH                     | test01                 | HEADER                     |
| test02                                  | HEADER                   | test05                 | PATH                       |
| test03                                  | QUERY                    | test03                 | HEADER                     |
| Backend request path: /apitest/{test05} |                          |                        |                            |

Assume that **test01** is **aaa**, **test02** is **bbb**, and **test03** is **ccc**.

The API request is as follows:

```
curl -ik -H 'test02:bbb' -X GET https://example.com/v1.0/aaa?test03=ccc
```

The backend service request is as follows:

```
curl -ik -H 'test01:aaa' -H 'test03:ccc' -X GET https://apitest/bbb
```

3. (Optional) Configure constant parameters of the default backend. Constant parameters can be defined to receive fixed constants. When sending a request to a backend service, ROMA Connect adds constant parameters to the specified location in the request and then sends the request to the backend service.

In the **Constant Parameters** area, click **Add Constant Parameter** to add the constant parameters of the backend service request.

**Table 7-9** Constant parameters

| Parameter   | Description  |
|-------------|--|
| Name        | Enter the name of a constant parameter. If <b>Location</b> is <b>PATH</b> , the parameter name must be the same as that in the backend request path.<br><b>NOTE</b> <ul style="list-style-type: none"><li>The parameter name is not case-sensitive. It cannot be <b>x-stage</b> or start with <b>x-api-</b> or <b>x-sdk-</b>.</li><li>If <b>Location</b> is <b>HEADER</b>, the parameter name is not case-sensitive.</li></ul> |
| Location    | Select the location of the constant parameter in the backend service request. The value can be <b>PATH</b> , <b>HEADER</b> , or <b>QUERY</b> .   |
| Value       | Enter the value of the constant parameter.   |
| Description | Enter the description of the constant parameter.   |

 **NOTE**

- ROMA Connect sends requests containing constant parameters to backend services after percent-encoding of special parameter values. Ensure that the backend services support percent-encoding. For example, parameter value **[api]** becomes **%5Bapi%5D** after percent-encoding.
  - For values of path parameters, ROMA Connect will percent-encode the following characters: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters `?></%#"[\]^`{}`
  - For values of query parameters, ROMA Connect will percent-encode the following characters: ASCII codes 0–31, blank symbols, ASCII codes 127–255, and special characters `>=<+&%#"[\]^`{}`
4. (Optional) Configure system parameters of the default backend. If a backend service needs to receive parameter information generated during system running, such as gateway built-in parameters, frontend authentication parameters, and backend authentication parameters, you can set system parameters. When sending a request to a backend service, ROMA Connect adds system parameters to the specified location in the request and then sends the request to the backend service.

In the **System Parameters** area, click **Add System Parameter** to add the system parameters of the backend service request.

**Table 7-10** System parameters

| Parameter             | Description   |
|-----------------------|---|
| System Parameter Type | <p>Select the type of a system parameter.</p> <ul style="list-style-type: none"><li>• <b>Default gateway parameter:</b> Parameters that can be configured for ROMA Connect.</li><li>• <b>Frontend authentication parameter:</b> Parameters to be displayed in the frontend custom authentication result. This option is available only if <b>Custom</b> is selected for <b>Security Authentication</b> in the step of <a href="#">setting basic information</a>.</li><li>• <b>Backend authentication parameter:</b> Parameters to be displayed in the backend custom authentication result. This option is available only if <b>Backend Authentication</b> is enabled in the step of <a href="#">defining backend request</a>.</li></ul>  |
| System Parameter Name | <p>Enter the name of a system parameter.</p> <ul style="list-style-type: none"><li>• If <b>System Parameter Type</b> is <b>Default gateway parameter</b>, select the parameters that can be obtained by the system.<ul style="list-style-type: none"><li>– <b>sourceIp:</b> source IP address of the client that calls an API.</li><li>– <b>stage:</b> name of the environment in which the API is published.</li><li>– <b>apiId:</b> ID of the API.</li><li>– <b>appId:</b> ID of the integration application used to call the API.</li><li>– <b>requestId:</b> request ID generated when the API is called.</li><li>– <b>serverAddr:</b> IP address of the gateway server.</li><li>– <b>serverName:</b> name of the gateway server.</li><li>– <b>handleTime:</b> processing time of the called API.</li><li>– <b>providerAppId:</b> ID of the integrated application to which the API belongs.</li><li>– <b>apiName:</b> name of the API. This parameter is available only after the API is published.</li><li>– <b>appName:</b> name of the integration application used to call the API.</li></ul></li><li>• If <b>System Parameter Type</b> is set to <b>Frontend authentication parameter</b> or <b>Backend authentication parameter</b>, you can define a value for <b>System Parameter Name</b>. However, the customized value must be set in the return result of custom authentication.</li></ul> |

| Parameter                  | Description   |
|----------------------------|---|
| Backend Parameter Name     | Enter the name of the backend parameter to be mapped.<br><b>NOTE</b> <ul style="list-style-type: none"> <li>The parameter name is not case-sensitive. It cannot be <b>x-stage</b> or start with <b>x-apig-</b> or <b>x-sdk-</b>.</li> <li>If <b>Backend Parameter Location</b> is <b>HEADER</b>, the parameter name is not case-sensitive.</li> </ul> |
| Backend Parameter Location | Select the location of the system parameter in the backend service request. The value can be <b>PATH</b> , <b>HEADER</b> , or <b>QUERY</b> .  |
| Description                | Enter the description of the system parameter.  |

5. (Optional) Add a backend policy. You can add multiple backend policies for an API as required and set different policy conditions to forward API requests to different backend services.
  - a. Click **Add Backend Policy** to add a backend policy for the API.
  - b. Configure information about the backend policy.

**Table 7-11** Parameters for creating a backend policy

| Parameter      | Description   |
|----------------|---|
| Name           | Enter a backend policy name to identify different backend policies.   |
| Effective Mode | Select the effective mode of the backend policy. <ul style="list-style-type: none"> <li><b>Any condition met:</b> If an API request meets any of the conditions in a policy, the request is forwarded to the backend.</li> <li><b>All conditions met:</b> API requests are forwarded to the backend only when all policy conditions are met.</li> </ul> |



| Parameter         | Description  |
|-------------------|--|
| Policy Conditions | <p>Add conditions for the backend policy to take effect.</p> <ul style="list-style-type: none"> <li>● <b>Condition:</b> conditions specified in the policy <ul style="list-style-type: none"> <li>- <b>Source IP address:</b> IP address from which the API is called</li> <li>- <b>Input parameter:</b> input parameter of an API request</li> <li>- <b>System parameter:</b> runtime parameter used by API Gateway to process API requests</li> <li>- <b>COOKIE:</b> cookies of an API request</li> </ul> </li> </ul> <p><b>NOTICE</b><br/>Input parameters (for example, headers) set as policy conditions must have already been defined in the API request settings.</p> <ul style="list-style-type: none"> <li>● <b>Parameter Name</b> <ul style="list-style-type: none"> <li>- When setting <b>Source</b> to <b>Input parameter</b>, select an input parameter.</li> <li>- When setting <b>Source</b> to <b>System parameter</b>, select a system parameter.<br/><b>reqPath:</b> Request URI, for example, <i>/a/b/c</i>.<br/><b>reqMethod:</b> Request method, for example, <b>GET</b>.</li> <li>- If <b>Source</b> is set to <b>COOKIE</b>, enter the parameter name.</li> </ul> </li> <li>● <b>Parameter Location:</b> The parameter location is displayed only if you set <b>Source</b> to <b>Input parameter</b>.</li> <li>● <b>Condition Type:</b> This parameter is mandatory only when you set <b>Condition</b> to <b>Input parameter</b>, <b>COOKIE</b>, or <b>System parameter</b>. <ul style="list-style-type: none"> <li>- <b>Equal:</b> The request parameter must be equal to the specified value.</li> <li>- <b>Enumerated:</b> The request parameter must be equal to any of the enumerated values.</li> <li>- <b>Matching:</b> The request parameter must be equal to any value of the regular expression.</li> </ul> </li> </ul> <p><b>NOTE</b><br/>When <b>Source</b> is set to <b>System parameter</b> and <b>Parameter Name</b> to <b>reqMethod</b>, you can set the condition type only to <b>Equal</b> or <b>Enumerated</b>.</p> <ul style="list-style-type: none"> <li>● <b>Condition Value:</b> value of the condition <ul style="list-style-type: none"> <li>- If <b>Condition Type</b> is set to <b>Equal</b>, enter a value.</li> <li>- If <b>Condition Type</b> is set to <b>Enumerated</b>, enter multiple values and separate them with commas (.).</li> <li>- If <b>Condition Type</b> is set to <b>Matching</b>, enter a regular expression, for example, <i>[0-5]</i>.</li> </ul> </li> </ul> |

| Parameter | Description  |
|-----------|--|
|           | <ul style="list-style-type: none"><li>- If <b>Source</b> is set to <b>Source IP address</b>, enter one or more IP addresses and separate them with commas (,).</li></ul> |

For example, there are three policy conditions whose **Source** is **Input Parameter**, as listed in the following table. If the request parameter value is **11**, policy A is met. If the request parameter value is **5**, policy B is met. If the request parameter value is **15**, policy C is met.

**Table 7-12** Policy parameters

| Policy   | Condition Type | Condition Value |
|----------|----------------|-----------------|
| Policy A | Equal          | 11              |
| Policy B | Enumerated     | 1, 2, 5, 8      |
| Policy C | Matching       | [0-5]           |

6. Click **Next** and **define the responses**.

## Defining Responses

1. Configure response examples to help API callers understand the responses to an API request.

**Table 7-13** Response configuration

| Parameter                | Description  |
|--------------------------|--|
| Example Success Response | Example of a successful response returned when the API is successfully called. |
| Example Failure Response | Example of a failure response returned when the API fails to be called.        |

2. Click **Finish**.

## 7.2.4 Debugging an API

### Overview

After creating an API, you can use the debugging function provided by ROMA Connect to debug the API.

## Prerequisites

- APIs with backend request paths containing variables cannot be debugged.
- During API debugging, the request throttling policy bound to the API becomes invalid.
- If the value of Timeout (ms) multiplied by the value of Retry Times is greater than 30 seconds **when defining the backend request**, the API debugging times out.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane, choose **API Connect > API Management**. On the **APIs** tab page, choose **More > Debug**.
3. In the API request parameter configuration area on the left of the page, configure API request information based on the API definition.

**Table 7-14** Request parameter configuration

| Parameter         | Description  |
|-------------------|--|
| Protocol          | This parameter can be set only if the API protocol is set to <b>HTTP&amp;HTTPS</b> .   |
| Method            | You can select a request method only if the API request method is set to <b>ANY</b> .  |
| Path              | You can add custom suffix (Suffix) to the request path only if the matching mode of the API request path is set to <b>Prefix match</b> . |
| Path Parameters   | This parameter can be only if the API request path contains the path parameter.  |
| Query Parameters  | Add and configure query parameters based on the API definition.  |
| Header Parameters | Add and configure header parameters based on the API definition.   |
| Body              | Body parameters can be configured only if <b>Method</b> is set to <b>POST, PUT, or PATCH</b> .   |

4. After setting the request parameters, click **Send Request**. The request sent by the API and the response received by the API are displayed in the right pane of the page.
  - If the API is successfully called, the status code 200 and a normal response body are returned.
  - If the API fails to be called, the status code 4xx or 5xx and error code description are returned. For details, see [Appendix: API Error Codes](#).You can send more requests with different parameters and values to verify the API.

## 7.2.5 (Optional) Creating the Environment and Environment Variables

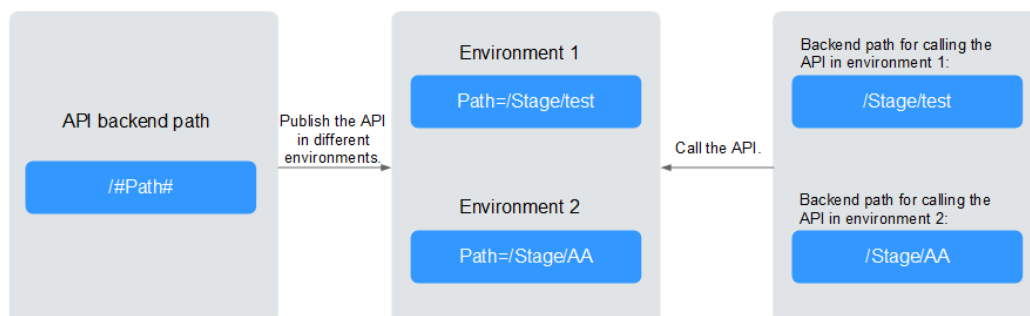
### Overview

An environment refers to the usage scope of an API. An API can be called only after it is published in an environment. APIs can be published in different customized environments, such as the development environment and test environment. RELEASE is the default and formal publishing environment of the system.

Environment variables are specific to environments. If environment variables are defined in backend information of an API, you need to add the variables to the environment. You can create variables in different environments to call different backend services using the same API.

For example, when an API is created, variable **Path** is defined in the backend request path. Two variables with the same name are respectively created and assigned values **/Stage/test** and **/Stage/AA** in environments 1 and 2. If the API is published and called in environment 1, the path **/Stage/test** is used. If the API is published and called in environment 2, the path **/Stage/AA** is used.

**Figure 7-3** Application of environment variables



### Creating an Environment

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Environments** tab page, click **Create**.
3. In the dialog box displayed, configure environment information and click **OK**.

**Table 7-15** Parameters for creating an environment

| Parameter   | Description  |
|-------------|--|
| Name        | Enter an environment name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Description | Enter a brief description of the environment.  |

When a user needs to call an open API, the API in the RELEASE environment is called by default. To access an API in another environment, add the **X-Stage** parameter to the header of the API request. The parameter value is the environment name. For example, to access an API in the Develop environment, add **X-Stage: Develop** to the header of the API request.

## Creating an Environment Variable

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **API Groups** tab page, click **More > Manage Variable** in the **Operation** column of the target API group.
3. Select the environment for which you want to add a variable and click **Create Variable**.
4. In the dialog box displayed, configure environment variable information and click **OK**.

**Table 7-16** Parameters for creating an environment variable

| Parameter | Description   |
|-----------|---|
| Name      | Enter the variable name, which must be the same as that defined in the API backend service information. |
| Value     | Enter the value of the environment variable.  |

## 7.2.6 Publishing an API

### Overview

After creating an API, you need to publish it to an environment so that it can be called by other users.

 **NOTE**

If a published API is modified, you must publish it again for the modifications to take effect in the environment.

### Prerequisites

The system provides the RELEASE environment by default. If you want to publish your API in another environment, [create an environment](#) first.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane, choose **API Connect > API Management**. On the **APIs** tab page, choose **Publish**.
3. On the **Publish API** page, configure publishing information and click **Publish**.

**Table 7-17** Parameters for publishing an API

| Parameter   | Description  |
|-------------|--|
| Environment | Select the environment in which the API is to be published. If your required environment is not available, click <b>Create Environment</b> on the right to create one. |
| Description | Enter the description of the API publication.  |

 **NOTE**

If the API has already been published in an environment, publishing it again will overwrite the existing one.

On the **APIs** tab page, click the API name to access the API details page. On the **Publication History** tab page, you can view the publishing history of the API and the API configuration in each version.

## 7.2.7 Binding Domain Names

### Overview

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are classified into private and public domain names.

- Private domain name: Service systems deployed on the cloud service platform can use private domain names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform can use public domain names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  - a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  - a. Apply for a public zone from Domain Registration.
  - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).

- c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

## Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **API Groups** tab page, click **More > Manage Domain Name** in the **Operation** column of the target API group.
3. On the **Domain Names** tab page, click **Bind Independent Domain Name**.
4. In the **Bind Independent Domain Name** dialog box displayed, enter the domain name and click **OK**.

**Table 7-18** Independent domain name configuration

| Parameter   | Description  |
|-------------|--|
| Domain Name | Enter the domain name to be bound.   |
| TLS Version | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate for the independent domain name. Otherwise, skip this step.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.  
If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

## 7.2.8 (Optional) Authorizing Applications to Call an API

### Overview

Applications need to be authorized to call APIs that use App authentication. When calling an API, a user gets authenticated using the key and secret of an integration application.

If **Simple Authentication** is enabled for an API, the AppCode configured in the integration application can be used for simple security authentication, and the key and secret do not need to be used for signature authentication.

#### NOTE


The integration application to which the API belongs can be directly used to call the API without authorization.

## Prerequisites

- The security authentication mode of the API is set to **App**.
- The API has been published in an environment. Otherwise, [publish the API](#) first.

## Granting Permissions for Integration Applications

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, select **Authorize** of an API.
3. On the **Authorize App** page, click **Select App**.
4. In the **Select App** dialog box, configure authorization information and click **OK**.

After the authorization is complete, click  on the left of the API to view the list of authorized integration applications.

**Table 7-19** Authorization configuration

| Parameter                | Description   |
|--------------------------|---|
| Environment              | Select the environment in which the API has been published.   |
| Integration Applications | Select the integration applications to be authorized for API calling.   |
| Access Parameters        | Set access parameters for the selected integration applications to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel            | Determine whether to enable the green channel. After the green channel is enabled, you also need to set <b>Whitelist</b> and <b>Blacklist</b> .   |
| Whitelist                | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. The IP addresses in the whitelist can call APIs without authentication.   |
| Blacklist                | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. The IP addresses in the blacklist are not allowed to call APIs.   |



## Adding AppCode

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Calling**. On the **Clients** tab page, click the name of the client (that is, the integration application) authorized by the API.
3. On the **AppCode** tab page of the client details page, click **Add AppCode**.
4. In the dialog box displayed, configure AppCode information and click **OK**.

**Table 7-20** AppCode configuration

| Parameter    | Description   |
|--------------|---|
| AppCode Type | Select the method for generating AppCode. <ul style="list-style-type: none"><li>• <b>Automatically generated:</b> AppCode is randomly generated by the system.</li><li>• <b>Custom:</b> Enter a user-defined AppCode.</li></ul> |
| AppCode      | Enter the value of AppCode.   |

## 7.3 Creating and Exposing Data APIs

### 7.3.1 Connecting to Data Sources

#### Overview

Before creating a data API, you need to connect to a data source to ensure that data can be read from the source. The data source connection configuration varies depending on data source types.

#### Prerequisites

- Before accessing a data source, ensure that the ROMA Connect instance can connect to the network where your data source resides.
  - Same VPC: The instance can directly access the data source.
  - Two VPCs in the same region: Connect the instance and the data source using a peering connection. For details, see [VPC Peering Connection](#).
  - Two VPCs in two regions: Connect the instance and the data source using a peering connection. For details, see [Network Communications Among VPCs Across Regions](#).
  - Communication over the public network: Ensure that the ROMA Connect instance has been bound with an EIP.
- To enable cross-VPC private access, [configure the routes](#) between the instance and the subnet where the data source resides.

## Connecting to a Data Source

- [Connecting to a DWS Data Source](#)
- [Connecting to a Gauss100 Data Source](#)
- [Connecting to a HANA Data Source](#)
- [Connecting to a HIVE Data Source](#)
- [Connecting to a MongoDB Data Source](#)
- [Connecting to a MySQL Data Source](#)
- [Connecting to an Oracle Data Source](#)
- [Connecting to a PostgreSQL Data Source](#)
- [Connecting to a Redis Data Source](#)
- [Connecting to a SQL Server Data Source](#)

### 7.3.2 Creating a Data API

#### Overview

ROMA Connect allows you to define a data source as a backend service so that data can be read from or written to the data source and exposed to external systems through APIs.

#### Prerequisites

- Data sources have been connected to ROMA Connect. For details, see [Connecting to Data Sources](#).
- If a backend service needs to use a signature key for authentication, [create a signature key](#) first.
- If you need to use the custom mode for API security authentication, [create a custom authorizer](#) first.
- Each line of record stored in a data source should not exceed 2 KB, or custom backend exceptions will occur.
- If a numeric value returned by the data source is zero with more than six decimal places, the data backend displays the value in scientific notation. Do not set the precision of numeric data to more than six decimal places.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. Create a backend.
  - a. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Create Backend**.
  - b. On the **Create Backend** page, set backend parameters and click **Create**.

**Table 7-21** Backend request information

| Parameter                       | Description  |
|---------------------------------|--|
| Name                            | Enter a backend name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Integration Application         | Select the integration application to which the backend belongs. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| Backend Request Method          | Select the request method of the backend. The value can be <b>GET</b> , <b>POST</b> , <b>PUT</b> , or <b>DELETE</b> .  |
| Backend Request Path            | Enter the request path of the backend, in the <code>/getUserInfo/userId</code> format.<br>The value of <b>Request Path</b> is case sensitive.  |
| Backend Security Authentication | Select the security authentication mode of the backend. <ul style="list-style-type: none"><li>● <b>Signature key</b>: A signature key is used to authenticate the request sender. If a signature key is used for authentication, the same signature key must be bound to the API that calls the backend service.</li><li>● <b>None</b>: No authentication is required for calling requests.</li></ul>  |
| Description                     | Enter a brief description of the backend.  |
| Advanced Settings               |  |
| Version                         | Enter the backend version, for example, <b>V1.0</b> . Custom backend version numbers differentiate the backend services.   |
| Input Parameters                | Define request parameters of the backend service.<br>In the <b>Input Parameters</b> area, click <b>Add Input Parameter</b> and add request parameters of the custom backend. <ul style="list-style-type: none"><li>● <b>Name</b>: name of a request parameter, which is user-defined.</li><li>● <b>Parameter Location</b>: location of the request parameter in the backend request. The value can be <b>Headers</b> or <b>Parameters</b>.</li><li>● <b>Default Value</b>: default value of the parameter, used only in the subsequent custom backend test procedure. This parameter does not take effect during custom backend deployment.</li><li>● <b>Mandatory</b>: whether a request parameter is mandatory in a backend request.</li><li>● <b>Description</b>: Enter the description of the parameter.</li></ul> |

| Parameter     | Description  |
|---------------|--|
| Returned Type | Select the response data format of the backend. The value can be <b>JSON</b> , <b>XML</b> , or <b>STREAM</b> . |
| Formatting    | This parameter specifies whether to format the response message body based on the selected return type.        |

After the backend is created, the online IDE of the backend is automatically displayed. The backend type is data backend by default.

3. Configure a data backend.
  - a. Click **Add Data Source** on the left of the online IDE.
  - b. On the **Add Data Source** page displayed, configure data source information and click **Add**.

**Table 7-22** Data source configuration

| Parameter             | Description  |
|-----------------------|--|
| Select Data Source    | Select a data source that you configured in <a href="#">Connecting to Data Sources</a> .   |
| Select Statement Type | Select the type of a statement to be executed. The value can be <b>SQL</b> or <b>SP</b> .<br>If the Redis or MongoDB data source is used, select <b>SQL</b> . The actual statement is NoSQL. |
| Advanced Settings     |  |
| Returned Object       | Enter the name of the returned object. The execution result of the statement is encapsulated in the object and returned.   |

| Parameter    | Description  |
|--------------|--|
| Paging       | <p>This parameter indicates whether the statement execution results are returned on multiple pages. If multiple data sources are added to the same data backend, you cannot set <b>Paging</b>.</p> <p>If paging is enabled, you can add the query parameters <b>pageNum</b> and <b>pageSize</b> to the backend request to page the query results and specify the page number of the data to be returned.</p> <ul style="list-style-type: none"> <li>• <b>pageNum</b>: page number of the data to be returned. The value starts from 1.</li> <li>• <b>pageSize</b>: number of data records on each page.</li> </ul> <p>The structure of the response result varies depending on whether paging is enabled or disabled. For details, see <a href="#">Example of Paging Results</a>.</p> <p><b>NOTE</b><br/>A maximum of 2000 data records can be obtained if paging is enabled. If there are more than 2000 records, try using the <b>offset</b> and <b>limit</b> parameters in the execution statement. If <b>Precompiling</b> is disabled, the following is an example:<br/>select * from table01 limit \${limit} offset \${offset}</p> <p>Values of the <b>offset</b> and <b>limit</b> parameters can be transferred in the headers, parameters, or body of backend requests.</p> <p>Data sources with precompiling enabled need to convert <b>offset</b> and <b>limit</b> values by calling functions. For details, see <a href="#">Developing Custom Data Backends</a>.</p> |
| Precompiling | This parameter specifies whether to precompile execution statements to prevent SQL injection risks.  |

### Example of Paging Results

If **Returned Object** is set to **mydata**, five data records are returned after the statement is executed to query data from the data source.

- If **Paging** is not enabled, all five data records are returned as the response result to the user. The following is an example of the response result.

```
{
  "mydata": [
    {
      "id": 1,
      "name": "aaa"
    },
    {
      "id": 2,
      "name": "bbb"
    },
    {
      "id": 3,
      "name": "ccc"
    },
    {
      "id": 4,
      "name": "ddd"
    }
  ]
}
```

```
},  
{  
  "id": 5,  
  "name": "eee"  
}  
]  
}
```

- When **Paging** is enabled, if **pageNum** is set to **1** and **pageSize** is set to **2**, the five data records queried by the statement will be displayed on multiple pages based on **pageSize**, with two data records on each page. Only the two data records on the first page will be returned to the user as the response result based on **pageNum**. In the response result, **total** indicates the total number of data records queried by running the statement, that is, **5**. Response example:

```
{  
  "mydata": {  
    "total": 5,  
    "data": [  
      {  
        "id": 1,  
        "name": "aaa"  
      },  
      {  
        "id": 2,  
        "name": "bbb"  
      }  
    ],  
    "pageSize": 2,  
    "pageNum": 1  
  }  
}
```

- c. After adding a data source, select the data source on the left of the online IDE, and then add an execution statement for the data source in the editing box on the right.  
For details, see [Developing Custom Data Backends](#).  
For the Redis or MongoDB data source, data processing commands of Redis or MongoDB are used.
  - d. Click **Save** in the upper right corner of the page to save the backend configuration.
4. Test the backend functions.  
In the upper right corner of the page, click **Test**. In the **Test Parameters** area, add request parameters based on the definition of the backend service and click **Test** to send the request.
    - In the **Execution Result** area, view the response of the backend.
    - In the **Execution History** area, view the historical test records of the backend. Click a test record to import historical test parameters to the test parameter list on the left and perform the test again.
  5. Deploy the backend.  
After the backend test is complete, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the backend.
  6. Publish a data API.
    - a. Click **Publish** in the upper right corner of the page.

- b. On the page displayed, configure API information and click **Publish** to create a frontend data API for the backend and publish the API in the environment.

**Table 7-23** Parameters for publishing an API

| Parameter                        | Description  |
|----------------------------------|--|
| API Group                        | Select an API group for the frontend API. If none is available, click <b>Create API Group</b> on the right to create one.<br><br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API.  |
| Environment                      | Select the environment in which the API is to be published. If your required environment is not available, click <b>Create Environment</b> on the right to create one.   |
| Frontend Security Authentication | Select the authentication mode of the API. The App authentication mode is recommended. <ul style="list-style-type: none"> <li>• <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an integration application.</li> <li>• <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK.</li> <li>• <b>Custom</b>: The custom function API is used for authenticating API requests.</li> <li>• <b>None</b>: Authentication is not required for API requests.</li> </ul> |
| Custom Authorizer                | This parameter is mandatory only if <b>Frontend Security Authentication</b> is set to <b>Custom</b> .<br>Select a frontend custom authorizer you have created.   |
| Frontend Request Protocol        | Select the request protocol used by the frontend API. The value can be <b>HTTP</b> , <b>HTTPS</b> , or <b>HTTP&amp;HTTPS</b> . <b>HTTPS</b> is recommended for transmitting important or sensitive data.   |
| Timeout (ms)                     | Enter the timeout interval of a backend service request. The default value is <b>60000</b> .   |

| Parameter               | Description  |
|-------------------------|--|
| Retry Times             | <p>Number of retry times after ROMA Connect fails to call the backend service.</p> <ul style="list-style-type: none"><li>• If the value is <b>-1</b>, the retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>• If the value ranges from 0 to 10, this parameter is enabled, and retries are performed based on the configured value.</li></ul>       |
| Advanced Settings       |  |
| Frontend Request Method | Select the request method of the frontend API. <b>ANY</b> indicates that the API can be accessed using any request method.   |
| Frontend Request Path   | <p>Enter the request path of the frontend API, for example, <b>/getUserInfo/userId</b>.</p> <p>The value of <b>Request Path</b> is case sensitive.</p>   |
| CORS                    | <p>This parameter specifies whether CORS is supported for the API.</p> <p>For security purposes, the browser restricts the cross-domain requests from being initiated from a page script. In this case, the page can access only the resources from the same source. CORS allows the browser to send XMLHttpRequest requests to the server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a>.</p> |

## 7.3.3 Binding Domain Names

### Overview

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are classified into private and public domain names.

- Private domain name: Service systems deployed on the cloud service platform can use private domain names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform can use public domain names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.



- a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
    - a. Apply for a public zone from Domain Registration.
    - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).
    - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

## Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **API Groups** tab page, click **More > Manage Domain Name** in the **Operation** column of the target API group.
3. On the **Domain Names** tab page, click **Bind Independent Domain Name**.
4. In the **Bind Independent Domain Name** dialog box displayed, enter the domain name and click **OK**.

**Table 7-24** Independent domain name configuration

| Parameter   | Description  |
|-------------|--|
| Domain Name | Enter the domain name to be bound.   |
| TLS Version | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate for the independent domain name. Otherwise, skip this step.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.  
If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

## 7.3.4 (Optional) Authorizing Apps to Call an API

### Overview

If a data API uses the App authentication mode, you need to authorize specified integration applications to call this API. When a user calls the API, the key and secret of the integration applications are used for authenticating API requests.

### Prerequisites

- The security authentication mode of the API is set to **App**.
- The API has been published in an environment. Otherwise, [publish the API](#) first.

### Granting Permissions for Integration Applications

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, select **Authorize** of an API.
3. On the **Authorize App** page, click **Select App**.
4. In the **Select App** dialog box, configure authorization information and click **OK**.

After the authorization is complete, click  on the left of the API to view the list of authorized integration applications.

**Table 7-25** Authorization configuration

| Parameter                | Description   |
|--------------------------|---|
| Environment              | Select the environment in which the API has been published.   |
| Integration Applications | Select the integration applications to be authorized for API calling.   |
| Access Parameters        | Set access parameters for the selected integration applications to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel            | Determine whether to enable the green channel. After the green channel is enabled, you also need to set <b>Whitelist</b> and <b>Blacklist</b> .   |
| Whitelist                | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. The IP addresses in the whitelist can call APIs without authentication.   |

| Parameter | Description   |
|-----------|---|
| Blacklist | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. The IP addresses in the blacklist are not allowed to call APIs. |

## 7.4 Creating and Exposing Function APIs

### 7.4.1 Creating a Function API

#### Overview

ROMA Connect allows you to define custom functions as backend services and open up function capabilities as APIs.

#### Prerequisites

- If a backend service needs to use a signature key for authentication, [create a signature key](#) first.
- If you need to use the custom mode for API security authentication, [create a custom authorizer](#) first.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. Create a backend.
  - a. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Create Backend**.
  - b. On the **Create Backend** page, set backend parameters and click **Create**.

**Table 7-26** Backend request information

| Parameter               | Description  |
|-------------------------|--|
| Name                    | Enter a backend name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Integration Application | Select the integration application to which the backend belongs. If none is available, click <b>Create Integration Application</b> on the right to create one. |
| Backend Request Method  | Select the request method of the backend. The value can be <b>GET</b> , <b>POST</b> , <b>PUT</b> , or <b>DELETE</b> .  |

| Parameter                       | Description  |
|---------------------------------|--|
| Backend Request Path            | Enter the request path of the backend, in the <code>/getUserInfo/userId</code> format.<br>The value of <b>Request Path</b> is case sensitive.  |
| Backend Security Authentication | Select the security authentication mode of the backend. <ul style="list-style-type: none"><li>● <b>Signature key</b>: A signature key is used to authenticate the request sender. If a signature key is used for authentication, the same signature key must be bound to the API that calls the backend service.</li><li>● <b>None</b>: No authentication is required for calling requests.</li></ul>  |
| Description                     | Enter a brief description of the backend.  |
| Advanced Settings               |  |
| Version                         | Enter the backend version, for example, <b>V1.0</b> . Custom backend version numbers differentiate the backend services.   |
| Input Parameters                | Define request parameters of the backend service.<br>In the <b>Input Parameters</b> area, click <b>Add Input Parameter</b> and add request parameters of the custom backend. <ul style="list-style-type: none"><li>● <b>Name</b>: name of a request parameter, which is user-defined.</li><li>● <b>Parameter Location</b>: location of the request parameter in the backend request. The value can be <b>Headers</b> or <b>Parameters</b>.</li><li>● <b>Default Value</b>: default value of the parameter, used only in the subsequent custom backend test procedure. This parameter does not take effect during custom backend deployment.</li><li>● <b>Mandatory</b>: whether a request parameter is mandatory in a backend request.</li><li>● <b>Description</b>: Enter the description of the parameter.</li></ul> |
| Returned Type                   | Select the response data format of the backend. The value can be <b>JSON</b> , <b>XML</b> , or <b>STREAM</b> .   |
| Formatting                      | This parameter specifies whether to format the response message body based on the selected return type.  |

After the backend is created, the online IDE of the backend is automatically displayed. The backend type is data backend by default.

### 3. Configure a function backend.

- a. In the upper left corner of the online IDE, choose **File > New Function Backend > Blank Template**. In the dialog box displayed, click **Yes** to switch the backend type to function backend.
- b. Edit the function script in the editing box on the right. You can also edit the function script using the script example provided by the system.

For details, see [Developing Custom Function Backends](#)

 **NOTE**

The maximum script size supported by a function API is 32 KB.

- c. Click **Save** in the upper right corner of the page to save the function backend configuration.
4. Test the backend functions.

In the upper right corner of the page, click **Test**. In the **Test Parameters** area, add request parameters based on the definition of the backend service and click **Test** to send the request.

    - In the **Execution Result** area, view the response of the backend.
    - In the **Execution History** area, view the historical test records of the backend. Click a test record to import historical test parameters to the test parameter list on the left and perform the test again.
  5. Deploy the backend.

After the backend test is complete, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the backend.
  6. Publish a function API.
    - a. Click **Publish** in the upper right corner of the page.
    - b. On the page displayed, configure API information and click **Publish** to create a frontend function API for the backend and publish the API in the environment.

**Table 7-27** Parameters for publishing an API

| Parameter   | Description   |
|-------------|---|
| API Group   | Select an API group for the frontend API. If none is available, click <b>Create API Group</b> on the right to create one.<br><b>NOTE</b><br>The API group cannot be changed once set and is bound to the access domain name of the API. |
| Environment | Select the environment in which the API is to be published. If your required environment is not available, click <b>Create Environment</b> on the right to create one.  |

| Parameter                        | Description   |
|----------------------------------|---|
| Frontend Security Authentication | Select the authentication mode of the API. The App authentication mode is recommended. <ul style="list-style-type: none"><li>● <b>App</b>: ROMA Connect authenticates API requests. When calling an API, a user gets authenticated using the key and secret of an integration application.</li><li>● <b>IAM</b>: IAM authenticates API requests. When calling an API, a user gets authenticated using the token or AK/SK.</li><li>● <b>Custom</b>: The custom function API is used for authenticating API requests.</li><li>● <b>None</b>: Authentication is not required for API requests.</li></ul> |
| Custom Authorizer                | This parameter is mandatory only if <b>Frontend Security Authentication</b> is set to <b>Custom</b> .<br>Select a frontend custom authorizer you have created.  |
| Frontend Request Protocol        | Select the request protocol used by the frontend API. The value can be <b>HTTP</b> , <b>HTTPS</b> , or <b>HTTP&amp;HTTPS</b> . <b>HTTPS</b> is recommended for transmitting important or sensitive data.  |
| Timeout (ms)                     | Enter the timeout interval of a backend service request. The default value is <b>60000</b> .  |
| Retry Times                      | Number of retry times after ROMA Connect fails to call the backend service. <ul style="list-style-type: none"><li>● If the value is <b>-1</b>, the retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>● If the value ranges from 0 to 10, this parameter is enabled, and retries are performed based on the configured value.</li></ul>   |
| Advanced Settings                |   |
| Frontend Request Method          | Select the request method of the frontend API. <b>ANY</b> indicates that the API can be accessed using any request method.  |
| Frontend Request Path            | Enter the request path of the frontend API, for example, <b>/getUserInfo/userId</b> .<br>The value of <b>Request Path</b> is case sensitive.  |

| Parameter | Description  |
|-----------|--|
| CORS      | <p>This parameter specifies whether CORS is supported for the API.</p> <p>For security purposes, the browser restricts the cross-domain requests from being initiated from a page script. In this case, the page can access only the resources from the same source. CORS allows the browser to send XMLHttpRequest requests to the server in a different domain. For details about CORS, see <a href="#">Configuring CORS for APIs</a>.</p> |

## 7.4.2 Binding Domain Names

### Overview

Before you expose an API, bind an independent domain name to the API group so that APIs in the group can be accessed with the domain name.

Independent domain names are classified into private and public domain names.

- Private domain name: Service systems deployed on the cloud service platform can use private domain names to access ROMA Connect APIs.
- Public domain name: Service systems deployed outside the cloud service platform can use public domain names to access ROMA Connect APIs.

For internal testing, use the default subdomain name to access APIs in an API group (maximum 1000 times a day). The subdomain name cannot be modified.

### Obtaining Domain Names

- To enable a service system on the cloud service platform to access APIs, obtain a private zone as an independent domain name.
  - a. Apply for a private zone. For details, see [Creating a Private Zone](#).
  - b. Configure an A record set to map the domain name to the APIC address. For details, see [Adding an A Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.
- To enable a service system outside the cloud service platform to access APIs, obtain a public zone as an independent domain name.
  - a. Apply for a public zone from Domain Registration.
  - b. Configure a CNAME record set to map the domain name to the APIC group subdomain name. For details, see [Adding a CNAME Record Set](#).
  - c. If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group. Obtain the content and keys of the SSL certificate and [create an SSL certificate](#) in advance.

## Binding Domain Names

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **API Groups** tab page, click **More > Manage Domain Name** in the **Operation** column of the target API group.
3. On the **Domain Names** tab page, click **Bind Independent Domain Name**.
4. In the **Bind Independent Domain Name** dialog box displayed, enter the domain name and click **OK**.

**Table 7-28** Independent domain name configuration

| Parameter   | Description  |
|-------------|--|
| Domain Name | Enter the domain name to be bound.   |
| TLS Version | Select the minimum TLS version used for domain name access. This parameter applies only to HTTPS. Other modes, such as HTTP, are not affected. |

5. (Optional) If the API group contains HTTPS-compatible APIs, bind an SSL certificate for the independent domain name. Otherwise, skip this step.
  - a. Click **Select SSL Certificate** on the right of the independent domain name.
  - b. In the displayed dialog box, select the SSL certificate to be bound and click **OK**.  
If no SSL certificate is available, click **Create SSL Certificate** to create one. For details, see [Creating an SSL Certificate](#).

### 7.4.3 (Optional) Authorizing Apps to Call an API

#### Overview

If the function API uses the App authentication mode, you need to authorize specified integration applications to call this API. When a user calls the API, the key and secret of the integration applications are used for authenticating API requests.

#### Prerequisites


- The security authentication mode of the API is set to **App**.
- The API has been published in an environment. Otherwise, [publish the API](#) first.

#### Granting Permissions for Integration Applications

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, select **Authorize** of an API.



3. On the **Authorize App** page, click **Select App**.
4. In the **Select App** dialog box, configure authorization information and click **OK**.

After the authorization is complete, click  on the left of the API to view the list of authorized integration applications.

**Table 7-29** Authorization configuration

| Parameter                | Description   |
|--------------------------|---|
| Environment              | Select the environment in which the API has been published.   |
| Integration Applications | Select the integration applications to be authorized for API calling.   |
| Access Parameters        | Set access parameters for the selected integration applications to be authorized. The access parameters will be added to the backend signature authentication information and sent to a backend service. The backend service then returns different response parameters based on the carried access parameters. |
| Green Channel            | Determine whether to enable the green channel. After the green channel is enabled, you also need to set <b>Whitelist</b> and <b>Blacklist</b> .   |
| Whitelist                | Mandatory only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the whitelist. The IP addresses in the whitelist can call APIs without authentication.   |
| Blacklist                | Available only when <b>Green Channel</b> is enabled.<br>Enter the IP addresses or IP address segments to be added to the blacklist. The IP addresses in the blacklist are not allowed to call APIs.   |

## 7.5 Calling an API

### 7.5.1 Calling an Open API

#### Overview

After an API is published in an environment, it can be called by other users. API calling operations vary by the authentication mode used by the API.

#### Prerequisites

Before calling an API, ensure that the network of your service system can communicate with the API access domain name or address.

- If the service system and the ROMA Connect instance are in the same VPC, the API can be directly accessed.
- Two VPCs in the same region: Connect the instance and the service system with a peering connection. For details, see [VPC Peering Connection](#).
- Two VPCs in two regions: Create a cloud connection and load the VPCs that need to communicate with each other. For details, see [Network Communications Among VPCs Across Regions](#).
- Communication over the public network: Ensure that the instance must be bound with an EIP.

## Obtaining API Calling Information

Obtain the API calling information from the API provider before you call an API.

- Obtaining API request information  
On the ROMA Connect instance console, choose **APIC > API Management**. On the **APIs** tab page, obtain the domain name, request method, and request path of an API. Click the API name to go to the details page. On the **API Calling** tab page, obtain the request protocol, input parameters, and request body description of the API.
- Obtaining API authentication information  
Obtain the request authentication information according to the authentication mode used by an API.

| Authentication Mode                                  | Authentication Info   |
|--|---|
| App authentication (with a signature)                | Obtain the key and secret of the integration application (or client AppKey and AppSecret) authorized by the API from the API provider, as well as the SDK, for authentication signatures. |
| App authentication (through simple authentication)   | Obtain the AppCode of the client authorized by the API from the API provider.   |
| App authentication (with green channel whitelisting) | Obtain the key (or client AppKey) of the integration application authorized for the API from the API provider.  |
| App authentication (with app_secret)                 | Obtain the key and secret (or client AppKey and AppSecret) of the integration application authorized for the API from the API provider.   |
| App authentication (with app_basic)                  | Obtain the key and secret (or client AppKey and AppSecret) of the integration application authorized for the API from the API provider.   |

| Authentication Mode               | Authentication Info   |
|-----------------------------------|---|
| App authentication (two-factor)   | Obtain the authentication information required for App authentication and custom authentication.            |
| IAM authentication (with a token) | Obtain the username and password of the cloud service platform.   |
| IAM authentication (with AK/SK)   | Obtain the AK/SK of the account on the cloud service platform and the SDK used for signatures.              |
| IAM authentication (two-factor)   | Obtain the authentication information required for IAM authentication and custom authentication.            |
| Custom authentication             | Obtain the custom authentication information to be carried in the request parameters from the API provider. |
| None                              | No authentication information is required.  |

- Obtaining the key and secret of an integration application  
On the ROMA Connect instance console, choose **Integration Applications**. Click the name of an integration application authorized by the API. On the details page that is displayed, obtain the key and secret of the application.
- Obtaining the AppKey and AppSecret of a client  
On the ROMA Connect instance console, choose **API Connect > API Calling**. On the **Clients** tab page, click the name of a client bound to the API. On the client details page that is displayed, obtain the AppKey and AppSecret of the client.
- Obtaining the SDK used for authentication signatures  
On the ROMA Connect instance console, choose **API Connect > API Calling**. Download the SDK of the required language on the SDK tab page.
- Obtaining the AppCode  
On the ROMA Connect instance console, choose **API Connect > API Calling**. On the **Clients** tab page, click the name of a client bound to the API. On the client details page that is displayed, obtain the AppCode on the AppCode tab page.

## Calling an API

1. Example API request  

```
POST https://{Address}/{Path}?{Query}
{Header}
{
```

```
{Body}
}
```

- *POST*: request method. Use the actual request method obtained in [Obtaining API request information](#).
- *{Address}*: request address. Use the actual request address obtained in [Obtaining API request information](#). You can also use an IP address to access an API.

| Scenario   | API Request Parameter Configuration  |
|--|--|
| Using a domain name to call an API                         | ROMA Connect allows APIs to be called using the subdomain name assigned to the API group or the domain name bound to the API group. No additional configuration is required.   |
| Calling an API in the DEFAULT group with an IP address     | Call an API in the DEFAULT group with an IP address. No additional configuration is required.  |
| Calling an API not in the DEFAULT group with an IP address | <ul style="list-style-type: none"> <li>• The <b>app_route</b> parameter described in <a href="#">Modifying Instance Configuration Parameters</a> has been set to <b>on</b> for a ROMA Connect instance, indicating that an API can be called by using an IP address.</li> <li>• ROMA Connect does not allow APIs in non-DEFAULT groups to be directly called using IP addresses. The header parameter <b>X-HW-ID</b> must be added to the request message, and the value must be the key or client AppKey of the integration application authorized by the API.</li> </ul> |

- *{Path}*: request path. Use the actual request path obtained in [Obtaining API request information](#).
  - (Optional) *{Query}*: query parameter in Parameter\_name=Parameter\_value format, for example, **limit=10**. Use & to separate multiple query parameters. For details, see the request parameters obtained in [Obtaining API request information](#).
  - *{Header}*: request header parameter in Parameter\_name:Parameter\_value format, for example, **Content-Type:application/json**. For details, see the request parameters obtained in [Obtaining API request information](#).
  - *{Body}*: request body in JSON format. For details, see the request body description obtained in [Obtaining API request information](#).
2. Add authentication information for the API request.

| API Authentication Mode                              | API Request Parameter Configuration  |
|--|--|
| App authentication (with a signature)                | Obtain the SDK to sign the API request. For details, see <a href="#">Developing API Calling Authentication (App)</a> .   |
| App authentication (through simple authentication)   | Add the header parameter <b>X-ApiG-AppCode</b> to the API request. The parameter value is the AppCode obtained in <a href="#">Obtaining API authentication information</a> .   |
| App authentication (with green channel whitelisting) | Add the header parameter <b>X-HW-ID</b> to the API request. The parameter value is the key of the integration application authorized for the API or the client's AppKey.   |
| App authentication (with app_secret)                 | <ul style="list-style-type: none"> <li>• The <b>app_secret</b> parameter has been set to <b>on</b> on the <a href="#">Configuration Parameters</a> tab page of a ROMA Connect instance, indicating that app_secret authentication is enabled.</li> <li>• Add the header parameter <b>X-HW-ID</b> to the API request. The parameter value is the key of the integration application authorized for the API or the client's AppKey.</li> <li>• Add the header parameter <b>X-HW-AppKey</b> to the API request. The parameter value is the secret or AppSecret obtained in <a href="#">Obtaining API authentication information</a>.</li> </ul> |
| App authentication (with app_basic)                  | <ul style="list-style-type: none"> <li>• The <b>app_basic</b> parameter described in <a href="#">Modifying Instance Configuration Parameters</a> has been set to <b>on</b> for a ROMA Connect instance, indicating that app_basic authentication is enabled.</li> <li>• Add the header parameter <b>Authorization</b> to the API request. The value is "<b>Basic "+base64(appkey +":"+appsecret)</b>". <b>appkey</b> and <b>appsecret</b> are the key and secret (or AppKey and AppSecret) obtained in <a href="#">Obtaining API authentication information</a>.</li> </ul>  |
| App authentication (two-factor)                      | An API request carries authentication information of both App authentication and custom authentication.  |
| IAM authentication (with a token)                    | Obtain the authentication token from the cloud service platform, add the header parameter <b>X-Auth-Token</b> to the API request, and set the value to the authentication token. For details, see <a href="#">Token Authentication</a> .   |

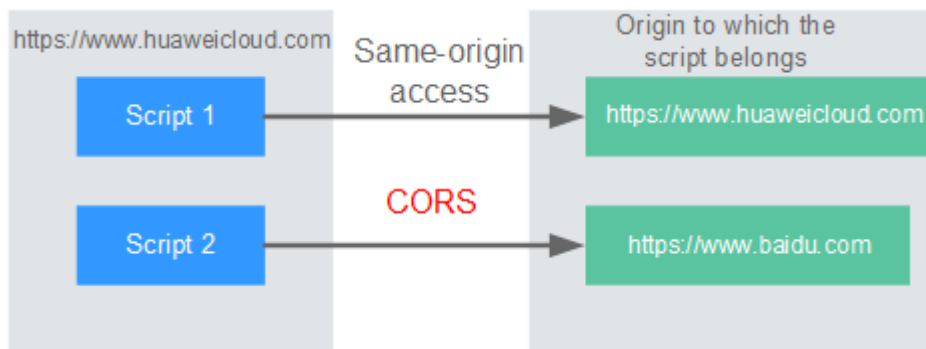
| API Authentication Mode         | API Request Parameter Configuration   |
|---------------------------------|---|
| IAM authentication (with AK/SK) | Sign API requests using the obtained SDK. For details, see <a href="#">AK/SK Authentication</a> .   |
| IAM authentication (two-factor) | An API request carries authentication information of both IAM authentication and custom authentication.   |
| Custom authentication           | Based on the definition of custom authentication, the related authentication information is carried in the API request parameters for authentication. |
| None                            | No authentication is required, and the API can be directly called.  |

## 7.5.2 Configuring CORS for APIs

### Overview

For security purposes, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send **XMLHttpRequest** requests to a server in a different domain.

Figure 7-4 Cross-domain access



CORS requests are classified into two types: simple requests and non-simple requests.

- **Simple requests** must meet both of the following conditions:
  - The request method is **HEAD**, **GET**, or **POST**.
  - The HTTP header can contain only the following fields: **Accept**, **Accept-Language**, **Content-Language**, **Last-Event-ID**, and **Content-Type** (only three values are allowed: **application/x-www-form-urlencoded**, **multipart/form-data**, and **text/plain**).

In the header of a simple request, browsers automatically add the **Origin** field to specify the origin (including the protocol, domain, and port) of the request.

After receiving such a request, the target server determines based on the origin whether the request is safe and can be accepted. If the server sends a response containing the **Access-Control-Allow-Origin** field, the server accepts the request.

- **Non-simple requests** do not meet the preceding two conditions.

Before sending a non-simple request, a browser first sends an HTTP request to the target server to determine whether the origin the web page is loaded from is in the allowed origin list, and which HTTP request methods and header fields can be used. Once the HTTP request is successfully preflighted, the browser then sends a simple request to the server.

By default, ROMA Connect does not support CORS. To use this feature, enable it when [creating an API](#). For a non-simple CORS request, you also need to create an API that uses the **OPTIONS** method for preflight.

## Simple Requests

- **Scenario 1:** If CORS is enabled and the response from the backend does not contain a CORS header, ROMA Connect can handle requests from any domain, and returns the Access-Control-Allow-Origin CORS header. The following messages are used as examples:

- a. **Request sent by a browser and containing the Origin header field:**

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

**Origin:** This field specifies the origin (<http://www.cors.com> in this example) of the request. It is mandatory. ROMA Connect and the backend service determine based on the origin whether the request is safe and can be accepted.

- b. **Response sent by the backend:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
```

```
{"status": "200"}
```

- c. **Response sent by ROMA Connect:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *
```

```
{"status": "200"}
```

**Access-Control-Allow-Origin:** This field is mandatory. The asterisk (\*) indicates that ROMA Connect accepts requests from any domain.

- **Scenario 2:** If CORS is enabled and the response from the backend contains a CORS header, the header will overwrite that added by ROMA Connect. The following messages are used as examples:

- a. **Request sent by a browser and containing the Origin header field:**

```
GET /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

**Origin:** This field specifies the origin (<http://www.cors.com> in this example) of the request. It is mandatory. ROMA Connect and the backend service determine based on the origin whether the request is safe and can be accepted.

b. **Response sent by the backend:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
Access-Control-Allow-Origin: http://www.cors.com
```

```
{"status":"200"}
```

**Access-Control-Allow-Origin:** Indicates that the backend service accepts requests sent from <http://www.cors.com>.

c. **Response sent by ROMA Connect:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: http://www.cors.com
```

```
{"status":"200"}
```

The CORS header in the backend response overwrites that in ROMA Connect's response.

## Non-Simple Requests

For a non-simple request, you also need to create an API using the **OPTIONS** method. The request parameters of an API accessed using the **OPTIONS** method must be set as follows:

- **Group:** The same group to which the API with CORS enabled belongs.
- **Security Authentication:** No authentication is required for requests received by the new API no matter which security authentication mode has been selected.
- **Protocol:** The same protocol used by the API with CORS enabled.
- **Request Path:** The same or matching request path used by the API with CORS enabled.
- **Method:** Select **OPTIONS**.
- **CORS:** Enabled.
- Backend service: **200 OK** is returned as the response.

The following are example requests and responses sent to or from a mock backend.

1. **Request sent from a browser to an API that is accessed using the OPTIONS method:**



```
OPTIONS /HTTP/1.1
User-Agent: curl/7.29.0
Host: localhost
Accept: */*
Origin: http://www.cors.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Sdk-Date
```

- **Origin:** This field is mandatory and used to specify the origin from which the request has been sent.
- **Access-Control-Request-Method:** This field is mandatory and used to specify the HTTP methods to be used by the subsequent simple requests.
- **Access-Control-Request-Headers:** This field is mandatory and used to specify the additional header fields in the subsequent simple requests.

## 2. Response sent by the backend:

None

## 3. Response sent by ROMA Connect:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 02:38:48 GMT
Content-Type: application/json
Content-Length: 1036
Server: roma
X-Request-Id: c9b8926888c356d6a9581c5c10bb4d11
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: X-Stage,X-Sdk-Date,X-Sdk-Nonce,X-Proxy-Signed-Headers,X-Sdk-Content-Sha256,X-Forwarded-For,Authorization,Content-Type,Accept,Accept-Ranges,Cache-Control,Range
Access-Control-Expose-Headers: X-Request-Id,X-Apig-Latency,X-Apig-Upstream-Latency,X-Apig-RateLimit-Api,X-Apig-RateLimit-User,X-Apig-RateLimit-App,X-Apig-RateLimit-Ip,X-Apig-RateLimit-Api-Allenv
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS,PATCH
Access-Control-Max-Age: 172800
```

- **Access-Control-Allow-Origin:** This field is mandatory. The asterisk (\*) indicates that ROMA Connect accepts requests from any domain.
- **Access-Control-Allow-Headers:** This field is required if it is contained in the request. It specifies the header information field supported by ROMA Connect.
- **Access-Control-Allow-Methods:** This field is mandatory and used to specify the HTTP request method supported by ROMA Connect.
- **Access-Control-Max-Age:** This field is mandatory and used to specify the period (in seconds) during which the preflight result remains valid. No more preflight requests are needed within the period.

## 4. Request sent by a browser and containing the Origin header field:

```
PUT /simple HTTP/1.1
Host: www.test.com
Origin: http://www.cors.com
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Accept: application/json
Date: Tue, 15 Jan 2019 01:25:52 GMT
```

**Origin:** This field is mandatory and used to specify the origin from which the request has been sent.

## 5. Response sent by the backend:

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
{"status":"200"}
```

**6. Response sent by APIC:**

```
HTTP/1.1 200 OK
Date: Tue, 15 Jan 2019 01:25:52 GMT
Content-Type: application/json
Content-Length: 16
Server: roma
X-Request-Id: 454d689fa69847610b3ca486458fb08b
Access-Control-Allow-Origin: *

{"status":"200"}
```

## 7.5.3 Viewing API Calling Statistics

### Overview

View API calling statistics on the ROMA Connect console.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Analysis**. On the **API Calling** tab page, view the API calling statistics, which include:
  - Real-time statistics on the total number of APIs, API groups, and request throttling policies
  - Number of requests, number of errors, data traffic, and call latency of an APIYou can also specify an integration application, API, and time range to view data.
  - Data in the last hour is updated every 2 minutes.
  - Data in the last 6 hours is updated every 2 hours.
  - Data in the last day is updated every 2 hours.
  - Data in the last week and last month is updated every day.

## 7.5.4 Viewing API Call Logs

### Overview

View API call logs on the ROMA Connect console.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Analysis**.
3. Enable log analysis.
  - a. On the **Access Logs** tab page, click **Configure Access Log**.
  - b. In the **Configure Access Log** dialog box, configure log access information and click **OK**.

**Table 7-30** Parameters for configuring access logs

| Parameter           | Description  |
|---------------------|--|
| Collect Access Logs | Determine whether to enable logging. API call logs can be viewed only after you enable <b>Collect Access Logs</b> .  |
| Log Group           | Select the log group to which the log stream belongs.<br>If no log group is available, click <b>View Log Group</b> to switch to the LTS console and create a log group. For details, see <a href="#">Creating a Log Group</a> .    |
| Log Stream          | Select the log stream for storing the API call logs.<br>If no log stream is available, click <b>View Log Stream</b> to switch to the LTS console and create a log stream. For details, see <a href="#">Creating a Log Stream</a> . |

- After the log analysis function is enabled, view the call logs of all open APIs in real time on the console. For details about the log fields, see [Log Field Description](#).

- In the upper right corner of the page, select a time segment.
- Click **View Log** to go to the LTS console to view log details and download logs locally.

Custom fields in log details:

- \_resource\_id**: ROMA Connect instance ID
- \_service\_type**: source service

## Log Field Description

| No. | Field          | Description   |
|-----|----------------|---|
| 1   | remote_addr    | Client IP address   |
| 2   | request_id     | Request ID  |
| 3   | api_id         | API ID  |
| 4   | user_id        | Project ID provided by a requester for IAM authentication.      |
| 5   | app_id         | Application ID provided by a requester using App authentication |
| 6   | time_local     | Request time  |
| 7   | request_time   | Request latency, in seconds                                     |
| 8   | request_method | HTTP request method   |
| 9   | host           | Request domain name   |

| No. | Field                      | Description  |
|-----|----------------------------|--|
| 10  | router_uri                 | Request URI  |
| 11  | server_protocol            | Request protocol   |
| 12  | status                     | Response code  |
| 13  | bytes_sent                 | Response size (including the status line, response header, and response body), in bytes  |
| 14  | request_length             | Request length (including the start line, request header, and request body), in bytes  |
| 15  | http_user_agent            | User agent ID  |
| 16  | http_x_forwarded_for       | X-Forwarded-For header field   |
| 17  | upstream_addr              | Backend address  |
| 18  | upstream_uri               | Backend URI  |
| 19  | upstream_status            | Backend response code  |
| 20  | upstream_connect_time      | Time taken for establishing a connection with the backend  |
| 21  | upstream_header_time       | Duration from the beginning of the establishment of a connection to receiving the first byte from the backend, in seconds  |
| 22  | upstream_response_time     | Duration from the beginning of the establishment of a connection to receiving the last byte from the backend, in seconds   |
| 23  | region_id                  | AZ ID  |
| 24  | all_upstream_response_time | Duration from the beginning of the establishment of a connection to receiving the last byte from the backend, in seconds. When a retry occurs, the value is the total time used. |
| 25  | errorType                  | Request error type. Options: <ul style="list-style-type: none"> <li>• 0: non-throttling error</li> <li>• 1: throttling error</li> </ul>  |
| 26  | auth_type                  | API authentication mode  |
| 27  | access_model1              | Authentication mode 1  |

| No. | Field                  | Description   |
|-----|------------------------|---|
| 28  | access_model2          | Authentication mode 2. When two-factor authentication is enabled, the custom authorizer ID is used.   |
| 29  | inner_time             | Internal processing duration of APIC, in seconds  |
| 30  | proxy_protocol_vni     | VPC endpoint virtual network ID   |
| 31  | proxy_protocol_vpce_id | VPC endpoint ID   |
| 32  | proxy_protocol_addr    | Client IP address   |
| 33  | body_bytes_sent        | Size of the API request body, in bytes  |
| 34  | api_name               | API name  |
| 35  | app_name               | Name of the application used by the requester when App authentication is used   |
| 36  | provider_app_id        | ID of the application to which the API belongs  |
| 37  | provider_app_name      | Name of the application to which the API belongs  |
| 38  | custom_data_log1       | Custom log field 1  |
| 39  | custom_data_log2       | Custom log field 2  |
| 40  | custom_data_log3       | Custom log field 3  |
| 41  | custom_data_log4       | Custom log field 4  |
| 42  | custom_data_log5       | Custom log field 5  |
| 43  | custom_data_log6       | Custom log field 6  |
| 44  | custom_data_log7       | Custom log field 7  |
| 45  | custom_data_log8       | Custom log field 8  |
| 46  | custom_data_log9       | Custom log field 9  |
| 47  | custom_data_log10      | Custom log field 10   |
| 48  | response_source        | Request response source. Options: <ul style="list-style-type: none"><li>• <b>local</b>: APIG</li><li>• <b>remote</b>: backend service</li></ul> |

## 7.5.5 Appendix: API Error Codes

The following table lists the error codes returned when a user fails to call an API.

**Table 7-31** Error codes

| HTTP Status Code | Error Code | Description  | Solution  |
|------------------|------------|--|---|
| 404              | APIC.0101  | The API does not exist or has not been published in the environment. | <ul style="list-style-type: none"><li>• Check whether the domain name, method, and path are consistent with those of the registered API.</li><li>• Check whether the API is published. If the API is published in a non-production environment, check whether the X-Stage header in the request is the name of the environment.</li></ul> |
| 500              | APIC.0103  | The backend does not exist.  | Contact technical support.  |
| 500              | APIC.0104  | The plug-ins do not exist.   | Contact technical support.  |
| 500              | APIC.0105  | The backend configurations do not exist.                             | Contact technical support.  |
| 400              | APIC.0106  | Orchestration error.   | Check whether the frontend and backend parameters of the API are correct.   |
| 400              | APIC.0201  | Bad request.   | Set valid request parameters.   |
| 413              | APIC.0201  | Request entity too large.  | Reduce the size of the request body to less than 12 MB.   |
| 414              | APIC.0201  | Request URI too large.   | Reduce the size of the URI to less than 32 KB.  |
| 494              | APIC.0201  | Request headers too large.   | Reduce the size of request headers to ensure that the length of a single request header is less than 32 KB or the total length of all request headers is less than 128 KB.  |
| 502              | APIC.0202  | Backend unavailable.   | Check whether the backend address configured for the API is accessible.   |
| 504              | APIC.0203  | Backend timeout.   | Increase the timeout duration of the backend service or shorten the processing time.  |

| HTTP Status Code | Error Code | Description   | Solution  |
|------------------|------------|---|---|
| 401              | APIC.0301  | Incorrect IAM authentication information.           | Check whether the token is correct.   |
| 403              | APIC.0302  | The IAM user is not authorized to access the API.   | Check whether the user is restricted by a blacklist or whitelist.   |
| 401              | APIC.0303  | Incorrect App authentication information.           | <ul style="list-style-type: none"> <li>• Check whether the request method, path, query parameters, and request body are consistent with those used for signature.</li> <li>• Check whether the client time is correct.</li> </ul> |
| 403              | APIC.0304  | The app is not authorized to access the API.        | Check whether the app has been authorized to access the API.  |
| 401              | APIC.0305  | Incorrect authentication information.               | Check whether the authentication information is correct.  |
| 403              | APIC.0306  | API access denied.                                  | Check whether you have been authorized to access the API.   |
| 401              | APIC.0307  | The token must be updated.                          | Update the token.   |
| 429              | APIC.0308  | The throttling threshold has been reached.          | Wait until the request throttling period ends and access the API again, or change the request throttling threshold.   |
| 403              | APIC.0401  | Unknown client IP address.                          | Contact technical support.  |
| 403              | APIC.0402  | The IP address is not authorized to access the API. | Check whether the IP address is restricted by a blacklist or whitelist.   |
| 503              | APIC.0404  | Access to the backend IP address has been denied.   | Use an available IP address to access the backend service.  |
| 403              | APIC.0405  | The app is not accessed from a trusted IP address.  | Check whether the IP address is restricted by the client access control policy.   |
| 500              | APIC.0601  | Internal server error.                              | Contact technical support.  |

| HTTP Status Code | Error Code | Description                                | Solution  |
|------------------|------------|--|---|
| 400              | APIC.0602  | Bad request.                               | Check whether the request is valid.   |
| 500              | APIC.0605  | Backend domain name resolution failed.     | Check whether the domain name is correct and has been bound to a correct backend address.                                 |
| 500              | APIC.0606  | Failed to load the API configurations.     | Contact technical support.  |
| 400              | APIC.0607  | The following protocol is supported: {xxx} | Use HTTP or HTTPS to access the API.  |
| 500              | APIC.0608  | Failed to obtain the admin token.          | Contact technical support.  |
| 500              | APIC.0609  | The VPC backend does not exist.            | Contact technical support.  |
| 502              | APIC.0610  | No backend available.                      | Check whether all backends are available.   |
| 500              | APIC.0611  | The backend port does not exist.           | Contact technical support.  |
| 500              | APIC.0612  | An API cannot call itself.                 | Modify the backend configurations, and ensure that the number of layers the API is recursively called does not exceed 10. |
| 500              | APIC.0705  | Backend signature calculation failed.      | Contact technical support.  |

## 7.5.6 Response Headers

The following table describes the response headers that APIC adds to the response returned when an API is called.

**X-Apig-Mode: debug** indicates APIC debugging information.

| Response Header | Description | Remarks                          |
|-----------------|-------------|----------------------------------|
| X-Request-Id    | Request ID. | Returned for all valid requests. |



| Response Header             | Description  | Remarks  |
|-----------------------------|--|--|
| X-Apig-Latency              | Duration from the time when APIC receives a request to the time when the backend returns a message header.             | Returned only when the request header contains <b>X-Apig-Mode: debug</b> .   |
| X-Apig-Upstream-Latency     | Duration from the time when APIC sends a request to the backend to the time when the backend returns a message header. | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and the backend type is not Mock.   |
| X-Apig-RateLimit-api        | API request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                                 | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called.                  |
| X-Apig-RateLimit-user       | User request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                                | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by a user.        |
| X-Apig-RateLimit-app        | Credential request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                          | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by a credential.  |
| X-Apig-RateLimit-ip         | IP address request limit information.<br>Example:<br><b>remain:9,limit:10,time:10 second.</b>                          | Returned only when the request header contains <b>X-Apig-Mode: debug</b> and a limit has been configured for the number of times the API can be called by an IP address. |
| X-Apig-RateLimit-api-allenv | Default API request limit information.<br>Example:<br><b>remain:199,limit:200,time:1 second.</b>                       | Returned only when the request header contains <b>X-Apig-Mode: debug</b> .   |
| X-Apig-count                | Total number of times that a request is forwarded by APIC.   | Returned for all valid requests forwarded by APIC. If the value of <b>X-Apig-count</b> is greater than 10, error APIC.0612 is reported.                                  |

## 7.6 Managing APIs

### 7.6.1 Taking an API Offline

#### Overview

If a published API is not needed to provide services, take it offline from its published environment.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, choose **More > Take Offline** of an API.
3. In the dialog box displayed, select the environment in which the API is to be taken offline and click **Yes**.

---

#### NOTICE

Taking an API offline will make the API inaccessible in the environment. Before performing this operation, ensure that you have notified the users who have used this API.

---

### 7.6.2 Importing and Exporting APIs

#### Overview

ROMA Connect allows you to import and export APIs using files.

- Importing APIs: API files in YAML and JSON formats can be imported, and the file content must comply with the Swagger 2.0 specifications.
- Exporting APIs: API files in YAML and JSON formats can be exported.

#### Prerequisites

- You have supplemented the [extended Swagger definition of APIs](#) in the API file to be imported.
- Ensure that the quotas of APIs and API groups meet the requirements before importing APIs.
- If you choose **New Group** when importing an API, the value of the **info.title** field in the API definition file is used as the API group name. Before importing an API, do not change the value of the **info.title** field.

#### Importing APIs

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.

2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, choose **More > Import API** above the API list.
3. On the **Import API** page, configure API import information.

**Table 7-32** Parameters for importing APIs

| Parameter                     | Description  |
|-------------------------------|--|
| Import To                     | <p>Select the method for importing APIs.</p> <ul style="list-style-type: none"> <li>• <b>New group:</b> Import APIs to a new API group. If you select this option, the system automatically creates an API group and imports the APIs to this group.</li> <li>• <b>Existing group:</b> Select an existing API group and add the imported APIs to this group.</li> </ul>  |
| Type                          | <p>This parameter is mandatory only if <b>Import To</b> is set to <b>New group</b>.</p> <p>Select the type of the API group.</p> <ul style="list-style-type: none"> <li>• <b>Integration application:</b> An API group belongs to a specific integration application. Only users who have permissions on the integration application can view and perform operations on the API group.</li> <li>• <b>Global:</b> All users can view and perform operations on the API group.</li> </ul>                                    |
| Integration Application       | <p>This parameter is mandatory only if <b>Type</b> is set to <b>Integration application</b>.</p> <p>Select the API group's integration application.</p>  |
| Overwrite APIs                | <p>This parameter is available only if <b>Import To</b> is set to <b>Existing group</b>.</p> <p>This parameter specifies whether to overwrite the existing APIs when the imported APIs conflict with those in the existing API group.</p>  |
| Overwrite Extended Definition | <p>This parameter specifies whether to overwrite the existing extended information, such as custom authentication, request throttling policy, and access control policy, when there is a conflict between the imported APIs and ROMA Connect.</p> <ul style="list-style-type: none"> <li>• <b>Enable:</b> The existing extended information is overwritten with that defined in the imported file.</li> <li>• <b>Disable:</b> The existing extended information is used, rather than that in the imported file.</li> </ul> |

| Parameter        | Description   |
|------------------|---|
| Parameter Import | <p>Select and check the imported file.</p> <ul style="list-style-type: none"><li>• Click <b>File</b> and select a local API file in YAML or JSON format.</li><li>• After the file is imported, click <b>Check</b> to check whether the format of the imported file is correct.</li><li>• Click <b>Format</b> to format the imported file.</li><li>• Click <b>Download</b> to download the imported file to the local host.</li><li>• Click <b>Enable Mock</b> if you want to use a mock backend for importing APIs.</li></ul> |

4. Click **Fast Import** to import APIs.

If you need to unify request and backend information of the imported APIs before the import, click **Configure Global Settings** and configure the information as prompted. Then click **Submit** to import the APIs.

If the imported APIs have not been published in the environment, publish them manually.

## Exporting APIs

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, choose **More > Export API** above the API list.
3. On the **Export API** page, configure API export information.

**Table 7-33** Parameters for exporting APIs

| Parameter   | Description  |
|-------------|--|
| API Group   | Select the API group from which APIs are to be exported.   |
| Environment | Select the environment in which the APIs to be exported have been published.   |
| APIs        | Select the APIs to be exported. If this parameter is not specified, all APIs in the API group in the selected environment will be exported by default. |

| Parameter      | Description  |
|----------------|--|
| API Definition | Select the scope of the API definition to be exported. <ul style="list-style-type: none"><li>• <b>Basic:</b> Only the API frontend request information is exported. The API frontend request information includes both standard and extended Swagger 2.0 fields.</li><li>• <b>Full:</b> Both API frontend request information and backend service information are exported.</li><li>• <b>Extended:</b> The API frontend request information, backend service information, as well as the request throttling policy and access control policy associated with the API are all exported.</li></ul> |
| Format         | Select the format of the exported API file. The value can be <b>YAML</b> or <b>JSON</b> .  |
| Version        | Enter the version of the API file to be exported. If no version is specified, the version will be set to the current time.   |

4. Click **Export** to export the API file to a local directory. The content of the exported file is displayed in the right pane.

 **NOTE**

- If no independent domain name is bound to the API group to which the exported API belongs, the subdomain name of the API group will be exported.
- If multiple independent domain names are bound to the API group to which the exported API belongs, only one random independent domain name will be exported.

## 7.6.3 Configuring an API Scheduled Task

### Overview

Create an API scheduled task on ROMA Connect so that APIs can be automatically called as scheduled.

### Prerequisites

- The API to be added to a scheduled task has been published. Otherwise, [publish the API](#) first.
- The API to be added to a scheduled task must use App or None authentication.
- [Instance parameters](#) **app\_route** and **app\_secret** of ROMA Connect have been set to **on**. That is, IP address access and app\_secret authentication are supported.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management > Scheduled Tasks**. On the page displayed, click **Create Scheduled Task**.

3. In the **Create Scheduled Task** dialog box, configure task information.

**Table 7-34** Task configuration

| Parameter               | Description  |
|-------------------------|--|
| Task Name               | Enter a task name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Integration Application | Select the integration application to which the task belongs. ROMA Connect will use the authentication information of the integration application to call APIs. If none is available, click <b>Create Integration Application</b> on the right to create one.  |
| Description             | Enter the description of the task.   |
| Effective Time          | Time when a scheduled task starts to be executed. The scheduled task will be executed based on the triggering condition after the execution time starts.   |
| Trigger Type            | The way to trigger the scheduled task. <ul style="list-style-type: none"><li>• <b>CRON</b>: The task is triggered using a cron expression.</li><li>• <b>CALENDAR</b>: The task is triggered at regular intervals.</li></ul>  |
| Cron Expression         | Mandatory for <b>Trigger Type</b> set to <b>CRON</b> .<br>Enter a cron expression to automatically trigger the task.   |
| Interval                | Mandatory for <b>Trigger Type</b> set to <b>CALENDAR</b> .<br>Enter the triggering period of the task. You can set the triggering period by minute, hour, day, week, or month.   |
| Max Retry Attempts      | Number of retry attempts after a task fails to be executed. The value ranges from 0 to 3.  |
| API                     | Click <b>Select API</b> on the right and select the API to be called by the task.<br><br>If integration applications of an API and a scheduled task are different, only an API of App authentication can be selected. You also need to <b>authorize the API</b> to access the integration application of the scheduled task. |

| Parameter          | Description  |
|--------------------|--|
| Request Parameters | <p>Add the request parameters for calling the API.</p> <ul style="list-style-type: none"><li>• <b>Path:</b> This parameter needs to be set only if the API request path contains the <b>Path</b> parameter. The system lists the required parameter by default.</li><li>• <b>Headers:</b> Configure the <b>Headers</b> parameter of the API request. Click <b>Add Request Parameter</b> and enter a parameter name and value.</li><li>• <b>Parameters:</b> Configure the query parameters of the API request. Click <b>Add Request Parameter</b> to add a parameter and enter a parameter name and value.</li><li>• <b>Body:</b> This parameter can be set only if the API request method is <b>POST</b>, <b>PUT</b>, or <b>PATCH</b>. Select a content type and enter the body content of the corresponding format as required.<ul style="list-style-type: none"><li>- <b>application/json:</b> data in JSON format</li><li>- <b>application/xml:</b> data in XML format</li><li>- <b>application/text:</b> data in text format</li></ul></li></ul> |

4. Click **OK**.
5. Start the API scheduled task.

A newly created task is in the **Initialized** state.

To start the task, click **Start** in the **Operation** column of the task. In the confirm dialog box displayed, click **Yes**. Then, the status of the task changes to **Started**, and the scheduled task will be executed based on the configured triggering mode.

## 7.6.4 SSL Certificate Management

### Overview

If the API group contains HTTPS-compatible APIs, add an SSL certificate for the independent domain name bound to the group.

### Prerequisites

- Only SSL certificates in PEM format can be added.
- The added SSL certificates support only the RSA, ECDSA, and DSA encryption algorithms.

### Creating an SSL Certificate

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **SSL Certificate Management** tab page, click **Create SSL Certificate**.
3. In the displayed dialog box, configure the SSL certificate information.

**Table 7-35** SSL certificate configuration

| Parameter | Description   |
|-----------|---|
| Name      | Enter an SSL certificate name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Scope     | Specify the scope to view the SSL certificate. <ul style="list-style-type: none"><li>● <b>Current instance:</b> The SSL certificate can be viewed only in the current instance.</li><li>● <b>All:</b> The SSL certificate can be viewed in all instances under the current account.</li></ul>                                   |
| Content   | Enter the SSL certificate content in PEM format.<br>Open the PEM certificate file in the certificate to upload in text, and copy the certificate content to <b>content</b> .<br>If the certificate is not in PEM encoding format, convert the format by referring to <a href="#">Converting the Certificate Format to PEM</a> . |
| Key       | Enter the SSL certificate key in PEM format.<br>Open the KEY/PEM private key file in the certificate to be uploaded in text, and copy the private key to <b>Key</b> .   |

4. Click **OK**. The SSL certificate is added.

## Converting the Certificate Format to PEM

| Format  | Converting with <a href="#">OpenSSL</a>   |
|---------|---|
| CER/CRT | Rename the certificate file <b>cert.crt</b> to <b>cert.pem</b> directly.  |
| PFX     | <ul style="list-style-type: none"><li>● Obtain a private key. For example, run the following command to convert <b>cert.pfx</b> into <b>key.pem</b>:<br/>openssl pkcs12 -in cert.pfx -nocerts -out key.pem</li><li>● Obtain a certificate. For example, run the following command to convert <b>cert.pfx</b> into <b>cert.pem</b>:<br/>openssl pkcs12 -in cert.pfx -nokeys -out cert.pem</li></ul>  |
| P7B     | <ol style="list-style-type: none"><li>1. Convert a certificate. For example, run the following command to convert <b>cert.p7b</b> into <b>cert.cer</b>:<br/>openssl pkcs7 -print_certs -in cert.p7b -out cert.cer</li><li>2. Rename the certificate file <b>cert.cer</b> to <b>cert.pem</b>.</li></ol>  |
| DER     | <ul style="list-style-type: none"><li>● Obtain a private key. For example, run the following command to convert <b>privatekey.der</b> into <b>privatekey.pem</b>:<br/><b>openssl rsa -inform DER -outform PEM -in privatekey.der -out privatekey.pem</b></li><li>● Obtain a certificate. For example, run the following command to convert <b>cert.cer</b> into <b>cert.pem</b>:<br/><b>openssl x509 -inform der -in cert.cer -out cert.pem</b></li></ul> |



## 7.6.5 Appendix: Extended Swagger Definition of APIs

On the basis of the basic Swagger definition, ROMA Connect defines the extended definition of APIs, such as the authentication mode and backend service definition. This section describes the extended definitions of APIs.

### 1: x-apigateway-auth-type

**Meaning:** Swagger-based apiKey authentication format, which defines an authentication mode provided by ROMA Connect.

**Scope of effect:** [Security Scheme Object](#)

#### Example

```
securityDefinitions:
  customize-name-iam:
    type: "apiKey"
    name: "unused"
    in: "header"
  x-apigateway-auth-type: "IAM"
  customize-name-app:
    type: "apiKey"
    name: "Authorization"
    in: "header"
  x-apigateway-auth-type: "AppSigv1"
  customize-name-iam-none:
    type: "apiKey"
    name: "unused"
    in: "header"
  x-apigateway-auth-type: "IAM_NONE"
```

**Table 7-36** Parameter description

| Parameter   | Mandatory | Type   | Description  |
|-------------|-----------|--------|--|
| type        | Yes       | String | Authentication type. Only <b>apiKey</b> is supported.  |
| name        | Yes       | String | Name of the parameter for authentication. <ul style="list-style-type: none"><li>When <b>x-apigateway-auth-type</b> is set to <b>AppSigv1</b>, set <b>name</b> to <b>Authorization</b>.</li><li>When <b>x-apigateway-auth-type</b> is set to <b>IAM</b> or <b>IAM_NONE</b>, set <b>name</b> to <b>unused</b>.</li></ul> |
| in          | Yes       | String | Location of the parameter. Only <b>header</b> is supported.  |
| description | No        | String | Description of the parameter.  |

| Parameter              | Mandatory | Type   | Description   |
|------------------------|-----------|--------|---|
| x-apigateway-auth-type | Yes       | String | APIC authentication mode. <b>AppSigv1</b> , <b>IAM</b> , and <b>IAM_NONE</b> are supported. |

## 2: x-apigateway-request-type

**Meaning:** API request type, which can be **public** or **private**.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-request-type: 'public'
```

**Table 7-37** Parameter description

| Parameter                 | Mandatory | Type   | Description  |
|---------------------------|-----------|--------|--|
| x-apigateway-request-type | Yes       | String | API visibility. The options include <b>public</b> and <b>private</b> . <ul style="list-style-type: none"> <li>• <b>public:</b> The API can be made available for sale.</li> <li>• <b>private:</b> The API will not be available for sale.</li> </ul> |

## 3: x-apigateway-match-mode

**Meaning:** Request URL matching mode, which can be **NORMAL** or **SWA**.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-apigateway-match-mode: 'SWA'
```

**Table 7-38** Parameter description

| Parameter               | Mandatory | Type   | Description  |
|-------------------------|-----------|--------|--|
| x-apigateway-match-mode | Yes       | String | Matching mode of the API request path. The value can be <b>SWA</b> or <b>NORMAL</b> . <ul style="list-style-type: none"> <li>• <b>SWA</b>: prefix match. For example, if the request path is set to <b>/test/AA</b> and the matching mode is set to <b>SWA</b>, the API can be accessed using <b>/test/AA/BB</b> or <b>/test/AA/CC</b> but cannot be accessed using <b>/test/AACC</b>.</li> <li>• <b>NORMAL</b>: exact match.</li> </ul> |

#### 4: x-apigateway-cors

**Meaning:** Specifies whether APIs defined by ROMA Connect support CORS.

**Scope of effect:** [Operation Object](#)

##### Example

```
paths:
  '/path':
    get:
      x-apigateway-cors: true
```

**Table 7-39** Parameter description

| Parameter         | Mandatory | Type    | Description                |
|-------------------|-----------|---------|----------------------------|
| x-apigateway-cors | No        | Boolean | Whether CORS is supported. |

For the API request for enabling CORS, the headers listed in the following table will be added to the response.

| Parameter Name         | Parameter Value | Description   |
|------------------------|-----------------|---|
| Access-Control-Max-Age | 172800          | Maximum time the response of a preflight request can be cached. |

| Parameter Name               | Parameter Value  | Description  |
|------------------------------|--|--|
| Access-Control-Allow-Origin  | *  | Domain that can be accessed. The asterisk (*) indicates that requests from any domain are allowed. |
| Access-Control-Allow-Headers | X-Sdk-Date, X-Sdk-Nonce, X-Proxy-Signed-Headers, X-Sdk-Content-Sha256, X-Forwarded-For, Authorization, Content-Type, Accept, Accept-Ranges, Cache-Control, Range | Header information fields that can be used in API requests.  |
| Access-Control-Allow-Methods | GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH   | HTTP methods allowed by the API request.   |

## 5: x-apigateway-any-method

**Meaning:** API request method used by default if no HTTP request method is specified.

**Scope of effect:** [Path Item Object](#)

### Example

```
paths:
  '/path':
    get:
      produces:
        - application/json
      responses:
        "200":
          description: "get response"
    x-apigateway-any-method:
      produces:
        - application/json
      responses:
        "200":
          description: "any response"
```

## 6: x-apigateway-backend

**Meaning:** API backend definition.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
```

```
description: "default response"
x-apigateway-request-type: "public"
x-apigateway-backend:
  type: "backend endpoint type"
```

**Table 7-40** Parameter description

| Parameter         | Mandatory | Type  | Description  |
|-------------------|-----------|---|--|
| type              | Yes       | String  | Backend service type. <b>HTTP</b> , <b>HTTP-VPC</b> , and <b>MOCK</b> are supported. |
| parameters        | No        | <b>x-apigateway-backend.parameters</b>        | Backend parameters.  |
| httpEndpoints     | No        | <b>x-apigateway-backend.httpEndpoints</b>     | HTTP backend service definition.   |
| httpVpcEndpoints  | No        | <b>x-apigateway-backend.httpVpcEndpoints</b>  | HTTP-VPC backend service definition.   |
| functionEndpoints | No        | <b>x-apigateway-backend.functionEndpoints</b> | Function backend service definition.   |
| mockEndpoints     | No        | <b>x-apigateway-backend.mockEndpoints</b>     | Mock backend service definition.   |

## 6.1: x-apigateway-backend.parameters

**Meaning:** API backend definition.

**Scope of effect:** **x-apigateway-backend**

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
```

```

    required: true
  - name: "userId"
    description: "user name"
    type: "string"
    in: "path"
    required: true
  responses:
    default:
      description: "default response"
  x-apigateway-request-type: "public"
  x-apigateway-backend:
    type: "HTTP"
    parameters:
      - name: "userId"
        value: "userId"
        in: "query"
        origin: "REQUEST"
        description: "user name"
      - name: "X-Invoke-User"
        value: "apigateway"
        in: "header"
        origin: "CONSTANT"
        description: "invoke user"

```

**Table 7-41** Parameter description

| Parameter   | Man<br>dator<br>y | Type   | Description   |
|-------------|-------------------|--------|---|
| name        | Yes               | String | Parameter name, which consists of a maximum of 32 bytes, starting with a letter. Only letters, digits, periods (.), hyphens (-), and underscores (_) are allowed.<br><br>The names of header parameters are not case-sensitive. |
| value       | Yes               | String | Parameter value, which is a parameter name if the parameter comes from a request.   |
| in          | Yes               | String | Parameter location, which can be <b>header, query, or path</b> .  |
| origin      | Yes               | String | Parameter mapping source. <b>REQUEST</b> and <b>CONSTANT</b> are supported.   |
| description | No                | String | Parameter description.  |

## 6.2: x-apigateway-backend.httpEndpoints

**Meaning:** HTTP backend service definition.

**Scope of effect:** **x-apigateway-backend**

### Example

```

paths:
  '/users/{userId}':

```

```

get:
  produces:
  - "application/json"
  parameters:
  - name: "X-Auth-Token"
    description: "authorization token"
    type: "string"
    in: "header"
    required: true
  responses:
  default:
    description: "default response"
  x-apigateway-request-type: "public"
  x-apigateway-backend:
    type: "HTTP"
  httpEndpoints:
    address: "www.example.com"
    scheme: "http"
    method: "GET"
    path: "/users"
    retryCount: "3"
    timeout: 30000

```

**Table 7-42** Parameter description

| Parameter  | Man<br>dator<br>y | Type   | Description  |
|------------|-------------------|--------|--|
| address    | Yes               | Array  | Backend service address. The format is <i>&lt;Domain name or IP address&gt;:[Port number]</i>                      |
| scheme     | Yes               | String | Backend request protocol. <b>HTTP</b> and <b>HTTPS</b> are supported.  |
| method     | Yes               | String | Backend request method. <b>GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH,</b> and <b>ANY</b> are supported.         |
| path       | Yes               | String | Backend request path, which can contain variables.   |
| retryCount | No                | String | Number of retry times upon a backend failure.  |
| timeout    | No                | Number | Backend request timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> . |

### 6.3: x-apigateway-backend.httpVpcEndpoints

**Meaning:** HTTP-VPC backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

**Example**

```

paths:
  '/users/{userId}':

```

```

get:
  produces:
    - "application/json"
  parameters:
    - name: "X-Auth-Token"
      description: "authorization token"
      type: "string"
      in: "header"
      required: true
  responses:
    default:
      description: "default response"
  x-apigateway-request-type: "public"
  x-apigateway-backend:
    type: "HTTP-VPC"
  httpVpcEndpoints:
    name: "vpc-test-1"
    scheme: "http"
    method: "GET"
    path: "/users"
    timeout: 30000
  
```

**Table 7-43** Parameter description

| Parameter | Man<br>dator<br>y | Type   | Description  |
|-----------|-------------------|--------|--|
| name      | Yes               | Array  | VPC channel name.  |
| scheme    | Yes               | String | Backend request protocol. <b>HTTP</b> and <b>HTTPS</b> are supported.  |
| method    | Yes               | String | Backend request method. <b>GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH,</b> and <b>ANY</b> are supported.         |
| path      | Yes               | String | Backend request path, which can contain variables.   |
| timeout   | No                | Number | Backend request timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> . |

## 6.4: x-apigateway-backend.functionEndpoints

**Meaning:** FUNCTION backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

### Example

```

paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
  
```



```

type: "string"
in: "header"
required: true
responses:
  default:
    description: "default response"
x-apigateway-request-type: "public"
x-apigateway-backend:
  type: "FUNCTION"
  functionEndpoints:
    version: "v1"
    function-urn: ""
    invocation-type: "synchronous"
    timeout: 30000
    
```

**Table 7-44** Parameter description

| Parameter       | Mandatory | Type   | Description   |
|-----------------|-----------|--------|---|
| function-urn    | Yes       | String | Function URN.   |
| version         | Yes       | String | Function version.   |
| invocation-type | Yes       | String | Function invocation type. <b>async</b> or <b>sync</b> are supported.  |
| timeout         | No        | Number | Function timeout in milliseconds. The value ranges from 1 to 60,000, and the default value is <b>5000</b> . |

## 6.5: x-apigateway-backend.mockEndpoints

**Meaning:** Mock backend service definition.

**Scope of effect:** [x-apigateway-backend](#)

### Example

```

paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      parameters:
        - name: "X-Auth-Token"
          description: "authorization token"
          type: "string"
          in: "header"
          required: true
      responses:
        default:
          description: "default response"
x-apigateway-request-type: "public"
x-apigateway-backend:
  type: "MOCK"
  mockEndpoints:
    result-content: "mocked"
    
```

**Table 7-45** Parameter description

| Parameter      | Man<br>dator<br>y | Type   | Description    |
|----------------|-------------------|--------|----------------|
| result-content | Yes               | String | Mock response. |

## 7: x-apigateway-backend-policies

**Meaning:** API backend policy.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
          x-apigateway-request-type: "public"
          x-apigateway-backend:
            type: "backend endpoint type"
          x-apigateway-backend-policies:
            - type: "backend endpoint type"
              name: "backend policy name"
            conditions:
              - type: "equal/enum/pattern",
                value: "string",
                origin: "source/request_parameter",
                parameter_name: "string"
```

**Table 7-46** Parameter description

| Parameter     | Man<br>dator<br>y | Type                                      | Description  |
|---------------|-------------------|---|--|
| type          | Yes               | String                                    | Backend service type. <b>HTTP</b> , <b>HTTP-VPC</b> , and <b>MOCK</b> are supported. |
| name          | Yes               | String                                    | Name   |
| parameters    | No                | <b>x-apigateway-backend.parameters</b>    | Backend parameters.  |
| httpEndpoints | No                | <b>x-apigateway-backend.httpEndpoints</b> | HTTP service definition.   |

| Parameter         | Mandatory | Type   | Description                  |
|-------------------|-----------|--|------------------------------|
| httpVpcEndpoints  | No        | <a href="#">x-apigateway-backend.httpVpcEndpoints</a>    | HTTP-VPC service definition. |
| functionEndpoints | No        | <a href="#">x-apigateway-backend.functionEndpoints</a>   | Function service definition. |
| mockEndpoints     | No        | <a href="#">x-apigateway-backend.mockEndpoints</a>       | Mock service definition.     |
| conditions        | Yes       | <a href="#">x-apigateway-backend-policies.conditions</a> | Backend policy condition.    |

## 7.1: x-apigateway-backend-policies.conditions

**Meaning:** API backend policy.

**Scope of effect:** [x-apigateway-backend-policies](#)

### Example

```
paths:
  '/users/{userId}':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
          x-apigateway-request-type: "public"
          x-apigateway-backend:
            type: "backend endpoint type"
          x-apigateway-backend-policies:
            - type: "backend endpoint type"
              name: "backend policy name"
          conditions:
            - type: "equal/enum/pattern",
              value: "string",
              origin: "source/request_parameter",
              parameter_name: "string"
```

**Table 7-47** Parameter description

| Parameter | Mandatory | Type   | Description   |
|-----------|-----------|--------|---|
| type      | Yes       | String | Policy condition type. <b>equal</b> , <b>enum</b> , and <b>pattern</b> are supported. |
| value     | Yes       | String | Policy condition value.   |
| origin    | Yes       | String | Policy condition source. <b>source</b> and <b>request</b> are supported.              |
| parameter | No        | String | Input parameter name if the <b>origin</b> parameter is set to <b>request</b> .        |

## 8: x-apigateway-ratelimit

**Meaning:** Request throttling policy.

**Scope of effect:** **Operation Object**

### Example

```
paths:
  '/path':
    get:
      x-apigateway-ratelimit: 'customRatelimitName'
```

**Table 7-48** Parameter description

| Parameter              | Mandatory | Type   | Description  |
|------------------------|-----------|--------|--|
| x-apigateway-ratelimit | No        | String | Name of the referenced request throttling policy. Set this parameter to <b>customRatelimitName</b> . |

## 9: x-apigateway-ratelimits

**Meaning:** Mapping between a request throttling policy name and limit values.

**Scope of effect:** **Swagger Object**

### Example

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: second/minute/hour
    shared: true
    special:
```

```
- type: APP
limit: 100
instance: xxxxxxxx
```

**Table 7-49** Parameter description

| Parameter           | Mandatory | Type   | Description   |
|---------------------|-----------|--|---|
| customRatelimitName | No        | <a href="#">x-apigateway-ratelimits.policy</a> | Custom request throttling policy. To use a request throttling policy, set <a href="#">x-apigateway-ratelimit</a> to the name of the policy. |

## 9.1: x-apigateway-ratelimits.policy

**Meaning:** Definition of a request throttling policy.

**Scope of effect:** [x-apigateway-ratelimits](#)

### Example

```
x-apigateway-ratelimits:
customRatelimitName:
api-limit: 200
app-limit: 200
user-limit: 200
ip-limit: 200
interval: 1
unit: MINUTE
shared: false
special:
- type: USER
limit: 100
instance: xxxxxxxx
```

**Table 7-50** Parameter description

| Parameter  | Mandatory | Type   | Description  |
|------------|-----------|--------|--|
| api-limit  | Yes       | Number | Maximum number of times an API can be called.  |
| user-limit | No        | Number | Number of times the API can be called by a user.   |
| app-limit  | No        | Number | Number of times the API can be called by an app.   |
| ip-limit   | No        | Number | Number of times the API can be called by an IP address.                                      |
| interval   | Yes       | Number | Throttling period.   |
| unit       | Yes       | String | Throttling unit, which can be <b>SECOND</b> , <b>MINUTE</b> , <b>HOURL</b> , or <b>DAY</b> . |

| Parameter | Mandatory | Type   | Description  |
|-----------|-----------|--|--|
| shared    | No        | Boolean  | Whether to share the throttling limits among APIs. |
| special   | No        | <a href="#">x-apigateway-ratelimits.policy.special</a> | Special request throttling policy.                 |

## 9.2: x-apigateway-ratelimits.policy.special

**Meaning:** Definition of a special request throttling policy.

**Scope of effect:** [x-apigateway-ratelimits.policy](#)

### Example

```
x-apigateway-ratelimits:
  customRatelimitName:
    api-limit: 200
    app-limit: 200
    user-limit: 200
    ip-limit: 200
    interval: 1
    unit: MINUTE
    shared: false
    special:
      - type: USER
        limit: 100
      instance: xxxxxxxx
```

**Table 7-51** Parameter description

| Parameter | Mandatory | Type   | Description  |
|-----------|-----------|--------|--|
| type      | Yes       | String | Excluded request throttling type, which can be <b>APP</b> or <b>USER</b> . |
| limit     | Yes       | Number | Access limit.  |
| instance  | Yes       | String | Excluded app or user   |

## 10: x-apigateway-access-control

**Meaning:** Access control policy.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
```

```
get:
  x-apigateway-access-control: 'customAccessControlName'
```

**Table 7-52** Parameter description

| Parameter                   | Mandatory | Type   | Description  |
|-----------------------------|-----------|--------|--|
| x-apigateway-access-control | No        | String | Name of the referenced access control policy. Set this parameter to <b>customAccessControlName</b> . |

## 11: x-apigateway-access-controls

**Meaning:** Mapping between an access control policy name and limit settings.

**Scope of effect:** [Swagger Object](#)

### Example

```
x-apigateway-access-controls:
  customAccessControlName:
    acl-type: "DENY"
    entity-type: "IP"
    value: 127.0.0.1,192.168.0.1/16
```

**Table 7-53** Parameter description

| Parameter               | Mandatory | Type  | Description   |
|-------------------------|-----------|---|---|
| customAccessControlName | No        | <a href="#">x-apigateway-access-controls.policy</a> | Custom access control policy. To use an access control policy, set <a href="#">x-apigateway-access-control</a> to the name of the policy. |

### 11.1: x-apigateway-access-controls.policy

**Meaning:** Definition of an access control policy.

**Scope of effect:** [x-apigateway-access-controls](#)

### Example

```
x-apigateway-access-controls:
  customAccessControlName:
    acl-type: "DENY"
    entity-type: "IP"
    value: 127.0.0.1,192.168.0.1/16
```

**Table 7-54** Parameter description

| Parameter   | Mandatory | Type   | Description  |
|-------------|-----------|--------|--|
| acl-type    | Yes       | String | Access control effect. The options include <b>PERMIT</b> and <b>DENY</b> . |
| entity-type | Yes       | String | Access control object. Only IP addresses are supported.                    |
| value       | Yes       | String | Access control policy values. Use commas (,) to separate multiple values.  |

## 12: x-apigateway-roma-app

**Meaning:** Integration application bound to the API.

**Scope of effect:** [Operation Object](#)

**Example**

```
paths:
  '/path':
    get:
      x-apigateway-roma-app: 'romaAppName'
```

**Table 7-55** Parameter description

| Parameter             | Mandatory | Type   | Description   |
|-----------------------|-----------|--------|---|
| x-apigateway-roma-app | Yes       | String | Name of the integration application bound to the API. |

## 13 x-apigateway-plugins

**Meaning:** API plug-in service.

**Scope of effect:** [Operation Object](#)

**Example:**

```
paths:
  '/path':
    get:
      x-apigateway-plugins: ['Plugin_mock']
```



**Table 7-56** Parameter description

| Parameter            | Mandatory | Type  | Description                        |
|----------------------|-----------|-------|------------------------------------|
| x-apigateway-plugins | No        | Array | List of plug-ins bound to the API. |

## 7.7 Managing Custom Backends

### 7.7.1 Taking a Custom Backend Offline

#### Overview

If you do not need a custom backend to provide services, take it offline. Its published API will be also taken offline and deleted.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Take Offline** in the **Operation** column of a backend.

When the backend status changes to **Developing**, the backend is taken offline.

---

**NOTICE**

Taking a backend offline will make the corresponding API inaccessible. Before performing this operation, ensure that you have notified the users who have used this API.

3. In the navigation pane on the left, choose **API Connect > API Management** to check whether the API corresponding to the backend has been deleted.

### 7.7.2 Importing and Exporting Custom Backends

#### Overview

ROMA Connect allows you to import and export custom backends using a file.

- Importing custom backends: API files in YAML and JSON formats can be imported, and the file content must comply with the Swagger 2.0 specifications.
- Exporting custom backends: API files in YAML and JSON formats can be exported.

## Prerequisites

- You have completed the [Swagger extended definitions for custom backends](#) in the API file to be imported.
- Before importing a custom backend, ensure that there is sufficient custom backend quota.
- When importing a custom backend, the maximum size of a target API definition file is 3 MB.
- When exporting a custom backend, the maximum size of a target API definition file is 50 MB. If the size of the target API definition file exceeds 50 MB, the excess part will not be exported.

## Importing Custom Backends

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Import** above the backend list.
3. On the **Import Backend** page, configure backend import information.

**Table 7-57** Parameters for importing backends

| Parameter                     | Description   |
|-------------------------------|---|
| Overwrite Basic Definition    | This parameter specifies whether to overwrite the existing backends when the imported backends conflict with the existing ones.   |
| Overwrite Extended Definition | This parameter specifies whether to overwrite the existing extended information, such as the request throttling policy and access control policy, when there is a conflict between the imported custom backends and ROMA Connect. <ul style="list-style-type: none"><li>• Enable: The existing extended information is overwritten with that defined in the imported file.</li><li>• Disable: The existing extended information is used, rather than that in the imported file.</li></ul> |
| Swagger                       | Click <b>Select</b> and select a local API file in YAML or JSON format.<br>After the file is imported, preview and modify the file online.  |

4. Click **Import**.  
The import result is displayed in the right pane. The **success** field contains the backends that are successfully imported, and the **failure** field contains the backends that fail to be imported, the error codes, and error information.

## Exporting Custom Backends

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.

- In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Export** above the backend list.  
Only backends in the **Developing** state can be exported.
- On the **Export Backend** page, configure backend export information.

**Table 7-58** Parameters for exporting backends

| Parameter               | Description   |
|-------------------------|---|
| API Definition          | Specify the type of API definition to be exported. Only <b>Full</b> is supported, indicating that all requests and service information of backends are exported.  |
| Scope                   | Specify the scope of custom backends to be exported. <ul style="list-style-type: none"><li><b>All</b>: All custom backends are exported.</li><li><b>Within integration application</b>: Custom backends of an integration application are exported.</li></ul> <b>NOTE</b><br>When you select <b>All</b> , only a random one of the custom backends with the same request path among different integration applications can be exported. |
| Integration Application | This parameter is mandatory only if <b>Scope</b> is set to <b>Within integration application</b> .<br>Select the integration application to which the custom backends belongs.  |
| Format                  | Select the format of the exported API file. The value can be <b>YAML</b> or <b>JSON</b> .   |

- Click **Export** to export the API file to a local directory. The content of the exported file is displayed in the right pane.

## 7.7.3 Adding Public Configurations for Custom Backends

### Overview

Global public configuration items, such as variables, passwords, and certificates, can be added for a custom backend. Besides, you can quickly reference the added configuration items in the JavaScript script of a function backend.

### Procedure

- Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
- In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Configurations** page, click **Add Configuration**.
- In the dialog box displayed, configure related information and click **OK**.

**Table 7-59** Public reference configurations

| Parameter               | Description   |
|-------------------------|---|
| Configuration Name      | Enter a configuration name.   |
| Integration Application | Select the integration application to which the configuration belongs.  |
| Configuration Type      | Select a configuration type. The value can be <b>Template variable</b> , <b>Password</b> , or <b>Certificate</b> .  |
| Configuration Value     | This parameter is available only when <b>Configuration Type</b> is set to <b>Template variable</b> or <b>Password</b> .<br>Enter the template variable or password.   |
| Confirm Value           | This parameter is available only when <b>Configuration Type</b> is set to <b>Password</b> .<br>Enter the password again, which must be the same as the value of <b>Configuration Value</b> .                        |
| Certificate             | This parameter is available only when <b>Configuration Type</b> is set to <b>Certificate</b> .<br>Enter the certificate in PEM format.  |
| Private Key             | This parameter is available only when <b>Configuration Type</b> is set to <b>Certificate</b> .<br>Enter the private key of the certificate in PEM format.   |
| Password                | This parameter is available only when <b>Configuration Type</b> is set to <b>Certificate</b> .<br>Enter the password of the certificate private key.  |
| Confirm Password        | This parameter is available only when <b>Configuration Type</b> is set to <b>Certificate</b> .<br>Enter the password of the certificate private key again, which must be the same as the value of <b>Password</b> . |
| Description             | Enter a description of the configuration.   |

4. Reference the configuration item in a backend function script.

For a configuration item named **example**, the reference format of each type of configuration is as follows:

- Template variable: `#{example}`
- Password: `CipherUtils.getPlainCipherText("example")`
- Certificate: `CipherUtils.getPlainCertificate("example")`

## 7.7.4 Appendix: Swagger Extended Definitions for Custom Backends

### Overview

Building on the basic Swagger definitions, ROMA Connect extends the definitions of APIs to the authentication mode and function script definition. This section describes the extended definitions for custom backends.

#### 1: x-livedata-auth-type

**Meaning:** Swagger-based apiKey authentication format, which defines an authentication mode provided by the custom backend.

**Scope of effect:** [Security Scheme Object](#)

#### Example

```
securityDefinitions:
  customize-name-signature:
    type: "apiKey"
    name: "Authorization"
    in: "header"
    x-livedata-auth-type: "SIGNATURE"
    x-livedata-signature:
      key: "signatureKey"
      secret: "signatureSecret"
```

**Table 7-60** Parameter description

| Parameter                   | Mandatory | Type   | Description   |
|-----------------------------|-----------|--------|---|
| type                        | Yes       | String | Authentication type. Only <b>apiKey</b> is supported.                                       |
| name                        | Yes       | String | Name of the parameter used for authentication. Set this parameter to <b>Authorization</b> . |
| in                          | Yes       | String | Location of the parameter. Only <b>header</b> is supported.                                 |
| description                 | No        | String | Description of the parameter.   |
| x-livedata-auth-type        | Yes       | String | Custom backend authentication mode. The value can only be <b>SIGNATURE</b> .                |
| x-livedata-signature.key    | No        | String | Key required for signature.   |
| x-livedata-signature.secret | No        | String | Secret required for signature.  |

## 2: x-livedata-version

**Meaning:** version number of the custom backend.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-livedata-version: '1.0.1'
```

**Table 7-61** Parameter description

| Parameter          | Mandatory | Type   | Description  |
|--------------------|-----------|--------|--------------|
| x-livedata-version | Yes       | String | API version. |

## 3: x-livedata-status

**Meaning:** Custom backend status.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-livedata-status: 'DESIGNED'
```

**Table 7-62** Parameter description

| Parameter         | Mandatory | Type   | Description   |
|-------------------|-----------|--------|---|
| x-livedata-status | Yes       | String | <p>Status of the custom backend. The options are <b>DESIGNED</b>, <b>DEVELOPED</b>, <b>TESTED</b>, and <b>DEPLOYED</b>.</p> <ul style="list-style-type: none"> <li>• <b>DESIGNED:</b> The custom backend has been designed and is to be developed.</li> <li>• <b>DEVELOPED:</b> The custom backend has been developed and is to be tested.</li> <li>• <b>TESTED:</b> The custom backend has been tested and is to be deployed.</li> <li>• <b>DEPLOYED:</b> The custom backend has been deployed.</li> </ul> |

## 4: x-livedata-roma-app

**Meaning:** Integration application bound to the custom backend.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      x-livedata-roma-app: 'romaAppName'
```

**Table 7-63** Parameter description

| Parameter           | Mandatory | Type   | Description  |
|---------------------|-----------|--------|--|
| x-livedata-roma-app | Yes       | String | Integration application bound to the custom backend. |

## 5: x-livedata-scripts

**Meaning:** custom backend definition script.

**Scope of effect:** [Operation Object](#)

### Example

```
paths:
  '/path':
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-livedata-scripts:
        - type: "function"
          content: "custom-script-content"
          result: "func"
```

**Table 7-64** Parameter description

| Parameter | Mandatory | Type   | Description   |
|-----------|-----------|--------|---|
| content   | Yes       | String | Script statement, which is a character string encoded using Base64. The actual script needs to be decoded using Base64. |

| Parameter   | Mandatory | Type   | Description   |
|-------------|-----------|--|---|
| result      | Yes       | String   | Return object. The execution result of the statement will be encapsulated in the object and returned.<br>This parameter is valid only for data backends, not function backends. |
| type        | Yes       | String   | Script type. The options are <b>Function</b> , <b>SQL</b> , and <b>SP</b> .   |
| datasources | No        | <a href="#">x-livodata-scripts.datasources</a> | Data source definition.   |

## 5.1 x-livodata-scripts.datasources

**Meaning:** data source definition of the custom backend.

**Scope of effect:** [x-livodata-scripts](#)

### Example

```
paths:
  /users:
    get:
      produces:
        - "application/json"
      responses:
        default:
          description: "default response"
      x-livodata-scripts:
        - type: "function"
          content: "custom-script-content"
          result: "custom-script-result"
          datasource:
            name: "custom-datasource-name"
```

**Table 7-65** Parameter description

| Parameter | Mandatory | Type   | Description       |
|-----------|-----------|--------|-------------------|
| name      | Yes       | String | Data source name. |

## 7.8 Managing Control Policies



## 7.8.1 Configuring a Request Throttling Policy

### Overview

Request throttling limits the number of times an API can be called within a period to protect the backend service. To provide continuous and stable services, create request throttling policies to control the number of calls made to your APIs.

A request throttling policy and an API are independent of each other. A request throttling policy takes effect for an API only after it is bound to the API.

#### NOTE

- An API can be bound to only one request throttling policy in an environment, but each request throttling policy can be bound to multiple APIs.
- If request throttling is triggered for an API, all API calling requests during the request throttling period will be discarded and a failure response will be returned to the calling party.

### Creating a Request Throttling Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Request Throttling Policies** tab page, click **Create**.
3. In the **Create Request Throttling Policy** dialog box, configure policy information.

**Table 7-66** Parameters for creating a request throttling policy

| Parameter | Description   |
|-----------|---|
| Name      | Enter a request throttling policy name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Type      | Select the type of the request throttling policy. <ul style="list-style-type: none"><li>• <b>API-specific:</b> Requests are throttled based on each API to which the policy is bound.</li><li>• <b>API-sharing:</b> Requests are throttled based on all APIs as a whole to which the policy is bound.</li></ul>   |
| Period    | Enter the request throttling duration. The unit can be seconds, minutes, hours, or days. This parameter must be used together with the following request throttling parameters: <ul style="list-style-type: none"><li>• <b>Max. API Requests</b> limits calls</li><li>• <b>Max. User Requests</b> limits calls by a user</li><li>• <b>Max. App Requests:</b> limits the total number of times an API can be called by an application within a period.</li><li>• <b>Max. IP Address Requests</b> limits calls by an IP address</li></ul> |

| Parameter                | Description  |
|--------------------------|--|
| Max. API Requests        | Enter the maximum number of times that an API can be called. This parameter is used along with <b>Period</b> .   |
| Max. User Requests       | Enter the maximum number of times that an API can be called by a user. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than that of <b>Max. API Requests</b> .         |
| Max. App Requests        | Enter the maximum number of times that an API can be called by an application. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than that of <b>Max. API Requests</b> . |
| Max. IP Address Requests | Enter the maximum number of times that an API can be called by an IP address. This parameter is used along with <b>Period</b> . The value of this parameter cannot be greater than that of <b>Max. API Requests</b> .  |
| Description              | Enter the descriptive information of the request throttling policy.  |

4. Click **OK**.

After the request throttling policy is created, you also need to perform the operations described in [Binding a Request Throttling Policy to an API](#) to make the policy take effect for the API.

## Binding a Request Throttling Policy to an API

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Request Throttling Policies** tab page, click **Bind to API** of a policy.
3. On the **Bind to API** page, click **Select API**.
4. In the **Select API** dialog box, select the APIs to which the request throttling policy is to be bound in the specified environment.  
APIs can be filtered by API group, environment, and API name.
5. Click **OK**.

## Binding a Request Throttling Policy to an Application

If you want to throttle requests for an integration application, you can add an excluded application to the request throttling policy. After an integration application is added to the request throttling policy, **Max. App Requests** of the application is restricted by the threshold of the excluded application, and **Max. API Requests** and **Max. User Requests** by the throttling policy.

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.

2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Request Throttling Policies** tab page, click the name of the request throttling policy that you want to bind to an application.
3. Click the **Excluded Apps** tab and click **Select Excluded App**.
4. In the **Select Excluded App** dialog box, configure application information.

**Table 7-67** Excluded app parameters

| Parameter | Description   |
|-----------|---|
| App       | Select the integration application to which the request throttling policy is to be bound.   |
| Threshold | Enter the maximum number of times that an API can be called by the integration application within a specified period. The value of this parameter cannot be greater than that of <b>Max. API Requests</b> in the request throttling policy. |

5. Click **OK**.

## Binding a Request Throttling Policy to a Tenant

If you want to throttle requests for a tenant, you can add an excluded tenant to the request throttling policy. After the tenant is added to the request throttling policy, **Max. User Requests** of the tenant is limited by the threshold of the excluded tenant, and **Max. API Requests** and **Max. App Requests** are limited by the throttling policy.

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Request Throttling Policies** tab page, click the name of the request throttling policy that you want to bind to an application.
3. Click the **Excluded Tenants** tab and click **Select Excluded Tenant**.
4. In the **Select Excluded Tenant** dialog box, configure tenant information.

**Table 7-68** Parameters for configuring an excluded tenant

| Parameter | Description  |
|-----------|--|
| Tenant ID | <p>Enter the ID of the tenant to which the request throttling policy is to be bound.</p> <ul style="list-style-type: none"><li>If the App authentication mode is used to call APIs, the tenant ID is the project ID of the user to which the integration application belongs.</li><li>If IAM authentication is used to call APIs, enter the account ID of the caller.</li></ul> <p>You can click the username in the upper right corner of the console and choose <b>My Credentials</b> to obtain the project ID and account ID of the user on the <b>My Credentials</b> page.</p> |
| Threshold | <p>Enter the maximum number of times that an API can be called by the tenant within a specified period. The value of this parameter cannot be greater than that of <b>Max. API Requests</b> in the request throttling policy.</p>  |

5. Click **OK**.

## 7.8.2 Configuring an Access Control Policy

### Overview

Access control allows you to control the IP addresses and accounts used to access APIs, protecting backend services. You can create an access control policy to allow or deny the access of certain IP addresses or accounts to an API.

An access control policy and an API are independent of each other. An access control policy takes effect for an API only after it is bound to the API.

#### NOTE

An API can be bound only to one access control policy of the same restriction type in an environment, but each access control policy can be bound to multiple APIs.

### Creating an Access Control Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Access Control Policies** tab page, click **Create**.
3. In the **Create Access Control Policy** dialog box, configure policy information.

**Table 7-69** Parameters for creating an access control policy

| Parameter        | Description   |
|------------------|---|
| Name             | Enter an access control policy name. It is recommended that you enter a name based on naming rules to facilitate search.  |
| Restriction Type | Select the restriction type of the access control policy. <ul style="list-style-type: none"><li>• <b>IP address</b>: restricts API calling by IP address.</li><li>• <b>Account name</b>: restricts API calling by account name. This option is available only to APIs using IAM authentication. The restriction also applies to the IAM users under the specified accounts. IAM users cannot be specified separately.</li></ul> |
| Effect           | Select the access control type. This parameter is used along with <b>Restriction Type</b> . <ul style="list-style-type: none"><li>• <b>Allow</b>: Only specified IP addresses or accounts are allowed to call APIs.</li><li>• <b>Deny</b>: Specified IP addresses or accounts are not allowed to call APIs.</li></ul>   |
| IP Address       | This parameter is mandatory only if <b>Restriction Type</b> is set to <b>IP address</b> .<br>Click <b>Add IP Address</b> to add the IP addresses or IP address segments that are allowed or forbidden to call an API.   |
| Account Name     | This parameter is mandatory only if <b>Restriction Type</b> is set to <b>Account name</b> .<br>Enter the account names that are allowed or forbidden to call an API. Use commas (,) to separate multiple account names.<br>You can click the username in the upper right corner of the console and choose <b>My Credentials</b> to obtain the account name of the user on the <b>My Credentials</b> page.                       |

4. Click **OK**.

After the access control policy is created, you also need to perform the operations described in [Binding an Access Control Policy to an API](#) to make the policy take effect for the API.

## Binding an Access Control Policy to an API

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Access Control Policies** tab page, click **Bind to API**.
3. On the **Bind to API** page, click **Select API**.

4. In the **Select API** dialog box, select the APIs to which the access control policy is to be bound in the specified environment.  
APIs can be filtered by API group, environment, and API name.
5. Click **OK**.

## 7.8.3 Configuring a Client Quota Policy

### Overview

A client quota limits the total number of times that an API can be called by a client in a specified period to protect backend services. You can create a client quota policy to limit the number of callings made by clients bound to the policy.

A client quota policy and a client are independent of each other. The client quota policy takes effect on the client only after the client is bound to the client quota policy.

Only users assigned with the **Tenant Administrator** permission can view and configure client quota policies.

#### NOTE

One client can be bound to only one client quota policy, but one client quota policy can be bound to multiple clients.

### Creating a Client Quota Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Calling**. On the **Client Quota Policies** tab page, click **Create**.
3. In the **Create Client Quota Policy** dialog box, configure policy information.

**Table 7-70** Parameters for creating a client quota policy

| Parameter    | Description   |
|--------------|---|
| Name         | Enter a client quota policy name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Effective On | Time when the quota policy takes effect. For example, if <b>Effective On</b> is set to <b>Aug 8, 2020 05:05:00</b> and <b>Period</b> is set to 1 hour, the quota policy took effect on Aug 8, 2020 05:05:00, and the period from the fifth minute of an hour to the fifth minute of the next hour is a cycle, for example, from 05:05:00 to 06:05:00. |
| Period       | Enter the period in which the quota policy is applied. The unit can be second, minute, hour, or day. This parameter must be used along with <b>Max. API Requests</b> to limit the total number of times an API can be called by a client within the specified period.   |

| Parameter         | Description  |
|-------------------|--|
| Max. API Callings | Enter the maximum number of times that an API can be called by a client. This parameter is used along with <b>Period</b> . |
| Description       | Enter a brief description of the client quota policy.  |

4. Click **OK**.

After the client quota policy is created, you also need to perform the operation described in [Binding a Quota Policy to a Client](#) to make the policy take effect for APIs.

## Binding a Quota Policy to a Client

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Calling**. On the **Client Quota Policies** tab page, click **Bind to Client** in the **Operation** column of a policy.
3. On the **Bind to Client** page, click **Select Client**.
4. In the **Select Client** dialog box, select the client to which the client quota policy is to be bound.

Search by client name is supported.

5. Click **Bind**.

A client can be bound to only one quota policy. If a client is bound to a quota policy repeatedly, the original quota policy will be unbound.

## 7.8.4 Configuring a Client Access Control Policy

### Overview

Client access control controls the client IP addresses that access APIs to protect backend services. You can configure an access control policy to allow or forbid a client with the specified IP address to access an API.

### Configuring an Access Control Policy

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Calling**. On the **Clients** tab page, click **Set Access Control** in the **Operation** column of a client.
3. In the **Set Access Control Policy** dialog box, configure policy information.

**Table 7-71** Parameters for setting an access control policy

| Parameter  | Description   |
|------------|---|
| Effect     | Access control type. Options: <ul style="list-style-type: none"><li>• <b>Allow</b>: Only clients with specified IP addresses are allowed to call APIs.</li><li>• <b>Deny</b>: Clients with specified IP addresses are not allowed to call APIs.</li></ul> |
| IP Address | Click <b>Add IP Address</b> to add the client IP addresses or IP address segments that are allowed or forbidden to call an API.   |

4. Click **OK**.

After configuring an access control policy, you can click **Reset** in the **Set Access Control Policy** dialog box to clear all access control policies.

## 7.9 Managing Plug-ins

### 7.9.1 Using Plug-ins

#### Overview

ROMA Connect provides flexible extension capabilities for APIs through plug-ins.

#### Prerequisites

- Plug-ins are independent of APIs. A plug-in takes effect for an API only after they are bound to each other. When binding a plug-in to an API, you must specify an environment where the API has been published. The plug-in takes effect for the API only in the specified environment.
- An API can be bound to only one plug-in of the same type in the same environment.
- Plug-ins that have been bound to APIs cannot be deleted.

#### Creating a Plug-in

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Plug-ins** tab page, click **Create Plug-in**.
3. In the **Create Plug-in** dialog box, configure the plug-in information.



**Table 7-72** Plug-in configuration

| Parameter               | Description   |
|-------------------------|---|
| Plug-in Name            | Name of the plug-in you want to create. It is recommended that you enter a name based on certain naming rules to facilitate identification and search.  |
| Plug-in Type            | Type of the plug-in, which determines the extension capabilities of the plug-in. <ul style="list-style-type: none"><li>• <b>CORS</b>: specifies preflight request headers and response headers and automatically creates preflight request APIs for cross-origin access.</li><li>• <b>Kafka Log Push</b>: pushes detailed API call logs to Kafka.</li><li>• <b>HTTP Response Header Management</b>: provides custom HTTP response headers and returns them in API responses.</li><li>• <b>Circuit Breaker</b>: protects the system when performance issues occur on backend services.</li></ul> |
| Plug-in Scope           | Specify the scope to view the plug-in. <ul style="list-style-type: none"><li>• <b>Integration application</b>: The plug-in belongs to an integration application. Only users who have the permission on the integration application can view and use the plug-in.</li><li>• <b>Global</b>: All users can view and use the plug-in.</li></ul>  |
| Integration Application | This parameter is mandatory only if <b>Plug-in Scope</b> is set to <b>Integration application</b> .<br>Select the integration application to which the plug-in belongs. If none is available, click <b>Create Integration Application</b> on the right to create one.   |
| Description             | Enter the plug-in description.  |
| Plug-in Content         | Content of the plug-in, which can be configured in a form or using a script.<br>The plug-in content varies depending on the plug-in type: <ul style="list-style-type: none"><li>• <b>CORS Plug-in</b></li><li>• <b>Kafka Log Push Plug-in</b></li><li>• <b>HTTP Response Header Management Plug-in</b></li><li>• <b>Circuit Breaker Plug-in</b></li></ul>   |

4. Click **OK**.

After a plug-in is created, perform the operations described in [Binding a Plug-in to an API](#) for the plug-in to take effect for the API.

## Binding a Plug-in to an API

1. In the navigation pane on the left, choose **API Connect > API Management**. On the **APIs** tab page, click the name of the API to which the plug-in is to be bound. The API details page is displayed.
2. On the **Plug-ins** tab page, click **Bind**.
3. In the **Bind Plug-in** dialog box, set **Environment** and **Plug-in Type**, and select the plug-in to be bound.  
You can search for a plug-in by name.
4. Click **OK**. The plug-in is bound to the API.

## 7.9.2 CORS Plug-in

### Overview

For security purposes, a browser restricts cross-domain requests initiated from scripts. That is, only resources from the same domain can be requested. However, CORS allows a browser to send **XMLHttpRequest** requests to a server in a different domain. For details about CORS, see [Configuring CORS for APIs](#).

The CORS plug-in provides the capabilities of specifying preflight request headers and response headers, as well as automatically creating preflight request APIs for quick and flexible cross-domain API access.

### Restrictions

- In the same API group, all APIs published in the same environment and with the same request path can be bound only to the same CORS plug-in.
- If you have enabled CORS for an API and have also bound the CORS plug-in to the API, the CORS plug-in will be used.
- If a request path contains an API with the **OPTIONS** method, none of the APIs in the request path can be bound to the CORS plug-in in the environment where the API is published.
- When you [bind a plug-in to an API](#), ensure that the request method of the API is included in **allow\_methods**.

### Parameter Description

Table 7-73 Configuration parameters

| Parameter     | Description  |
|---------------|--|
| allow origin  | Access-Control-Allow-Origin response header, which specifies the external domain URIs that are allowed to access the API. Use commas (,) to separate multiple URIs.<br>For requests that do not carry identity credentials, set this parameter to * to allow access requests from all domains. |
| allow methods | Access-Control-Allow-Methods response header, which specifies the allowed HTTP request methods. Use commas (,) to separate multiple request methods.   |

| Parameter         | Description   |
|-------------------|---|
| allow headers     | <p><b>Access-Control-Allow-Headers</b> response header, which specifies request headers that can be used when sending <b>XMLHttpRequest</b> requests. Use commas (,) to separate multiple headers.</p> <p>By default, simple request headers <b>Accept</b>, <b>Accept-Language</b>, <b>Content-Language</b>, and <b>Content-Type</b> (only if the value is <b>application/x-www-form-urlencoded</b>, <b>multipart/form-data</b>, or <b>text/plain</b>) are carried in requests. You do not need to configure these headers in this parameter.</p> |
| expose headers    | <p><b>Access-Control-Expose-Headers</b> response header, which specifies which response headers can be contained in the response of <b>XMLHttpRequest</b>. Use commas (,) to separate multiple headers.</p> <p>By default, basic response headers <b>Cache-Control</b>, <b>Content-Language</b>, <b>Content-Type</b>, <b>Expires</b>, <b>Last-Modified</b>, and <b>Pragma</b> can be contained in the response. You do not need to configure these headers in this parameter.</p>   |
| max age           | <p><b>Access-Control-Max-Age</b> response header, which specifies the validity period (in seconds) of the preflight request. No more preflight requests are needed within the period.</p>   |
| allow credentials | <p><b>Access-Control-Allow-Credentials</b> response header, which specifies whether <b>XMLHttpRequest</b> requests can carry cookies. Options:</p> <ul style="list-style-type: none"><li>• <b>true</b>: allowed</li><li>• <b>false</b>: not allowed</li></ul>   |

## Script Configuration Example

```
{
  "allow_origin": "*",
  "allow_methods": "GET,POST,PUT",
  "allow_headers": "Accept-Ranges,Cache-Control",
  "expose_headers": "X-Request-Id,X-Apig-Latency",
  "max_age": 172800,
  "allow_credentials": true
}
```

## 7.9.3 Kafka Log Push Plug-in

### Overview

ROMA Connect can collect call logs of open APIs in APIC. This plug-in pushes API call logs to Kafka so that users can easily obtain them.

## Constraints

- An API can be bound to only one plug-in policy of the same type in the same environment. A plug-in policy that has been bound to an API cannot be deleted.
- A maximum of five Kafka log push plug-ins can be created on a ROMA Connect instance.
- In the log data to be pushed, the response data does not support **Transfer Encoding** response header parameters.
- By default, a maximum of 4 KB of logs can be pushed. Excess logs will be truncated.
- The request body and response body in pushed logs are calculated in UTF-8 bytes.

## Parameter Description

**Table 7-74** Configuration parameters

| Parameter                 | Description  |
|---------------------------|--|
| <b>Policy Information</b> |  |
| Broker Addresses          | Connection addresses of target Kafka brokers to which logs are to be pushed. Use commas (,) to separate multiple addresses.  |
| Topic                     | Topic of the target Kafka to report logs to.   |
| Key                       | Enter the key value of a message so that the message will be stored in a specified partition of Kafka. It can be used as an ordered message queue. If this parameter is left empty, messages are stored in different message partitions in a distributed manner.   |
| Retry                     | Configuration for retrying when logs fail to be pushed to Kafka. <ul style="list-style-type: none"><li>• <b>Retry Times:</b> the number of retry attempts in case of a failure. Enter 0 to 5.</li><li>• <b>Retry Interval:</b> the interval of retry attempts in case of a failure. Enter 1 to 10 seconds.</li></ul> |
| <b>SASL Configuration</b> |  |
| Security Protocol         | Protocol used for connecting to the target Kafka. <ul style="list-style-type: none"><li>• <b>PLAINTEXT:</b> user authentication protocol of the default access point.</li><li>• <b>SASL_PLAINTEXT:</b> SASL user authentication protocol.</li><li>• <b>SASL_SSL:</b> SSL user authentication protocol.</li></ul>     |
| Message Tx/Rx Mechanism   | Message transmission and receiving mechanism of the target Kafka. The default value is <b>PLAIN</b> .  |

| Parameter                     | Description   |
|-------------------------------|---|
| SASL Username                 | Mandatory for <b>Security Protocol</b> set to <b>SASL_PLAINTEXT</b> or <b>SASL_SSL</b> .<br>Username used for SASL or SSL authentication.   |
| SASL Password                 | Mandatory for <b>Security Protocol</b> set to <b>SASL_PLAINTEXT</b> or <b>SASL_SSL</b> .<br>Password used for SASL or SSL authentication.   |
| Certificate Content           | Available for <b>Security Protocol</b> set to <b>SASL_SSL</b> .<br>CA certificate used for SSL authentication.  |
| <b>Metadata Configuration</b> |   |
| System Metadata               | System fields that need to be carried in pushed logs.<br>By default, the <b>start_time</b> , <b>request_id</b> , <b>client_ip</b> , <b>request_time</b> , <b>http_status</b> , <b>scheme</b> , <b>request_method</b> , <b>host</b> , <b>uri</b> , <b>upstream_addr</b> , <b>upstream_status</b> , <b>upstream_response_time</b> , <b>http_x_forwarded_for</b> , <b>http_user_agent</b> , and <b>error_type</b> fields are carried in logs. You can also specify other system fields that need to be included.   |
| Request Data                  | API request information that needs to be carried in the pushed logs. <ul style="list-style-type: none"> <li>● <b>The log contains the request header:</b> Specify a header that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li> <li>● <b>The log contains the request QueryString:</b> Specify a query string that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li> <li>● <b>The log contains the request body:</b> If this option is selected, logs will contain the body of API requests.</li> </ul> |
| Response Data                 | API response information that needs to be included in pushed logs. <ul style="list-style-type: none"> <li>● <b>The log contains the response header:</b> Specify a header that needs to be included. Use commas (,) to separate multiple fields. The asterisk (*) can be used as a wildcard.</li> <li>● <b>The log contains the response body:</b> If this option is selected, logs will contain the body of API request responses.</li> </ul>  |

| Parameter             | Description   |
|-----------------------|---|
| Custom Authentication | Custom authentication information that needs to be included in pushed logs. <ul style="list-style-type: none"><li>• <b>Frontend:</b> Enter a response field of frontend authentication that needs to be included. Separate multiple fields with commas (,).</li><li>• <b>Backend:</b> Enter a response field of backend authentication that needs to be included. Separate multiple fields with commas (,).</li></ul> |

## 7.9.4 HTTP Response Header Management Plug-in

### Overview

An HTTP response header is a part of an API response. An HTTP response header management plug-in enables you to customize HTTP response headers that will be returned in an API response.

### Restrictions

- The system response headers (such as **x-apig-\*** and **x-request-id**) cannot be modified.
- The response headers for CORS cannot be modified.

### Parameter Description

**Table 7-75** Configuration parameters

| Parameter | Description   |
|-----------|---|
| Name      | Response header name, which is case-insensitive and must be unique within a plug-in.  |
| Value     | Value of the response header. This parameter does not take effect and can be left blank if you set <b>Action</b> to <b>Delete</b> . |

| Parameter | Description  |
|-----------|--|
| Action    | <p>Response header operation. You can override, append, delete, skip, or add the specified header.</p> <ul style="list-style-type: none"> <li>● <b>Override</b> <ul style="list-style-type: none"> <li>- The value of this response header will override that of the same header that exists in an API response.</li> <li>- If an API response contains multiple response headers with the same name, only the value of this response header will be returned.</li> <li>- If there is no response header with the same name in an API response, the value of this response header will be returned.</li> </ul> </li> <li>● <b>Append</b> <ul style="list-style-type: none"> <li>- If an API response contains the specified header, the value you set here will be added, following the existing value. The two values will be separated with commas (,).</li> <li>- If an API response contains multiple headers with the same name as the one you set here, values of these headers will be separated with commas (,) and followed by the value of the specified header.</li> <li>- If there is no response header with the same name in an API response, the value of this response header will be returned.</li> </ul> </li> <li>● <b>Delete</b> <ul style="list-style-type: none"> <li>- This response header will be deleted if a response header with the same name exists in an API response.</li> <li>- If an API response contains multiple response headers with the same name, all these response headers will be deleted.</li> </ul> </li> <li>● <b>Skip</b> <ul style="list-style-type: none"> <li>- If an API response contains the specified header, the header will be skipped.</li> <li>- If an API response contains multiple headers with the same name as the one you set here, values of all these headers will be returned without modification.</li> <li>- If there is no response header with the same name in an API response, the value of this response header will be returned.</li> </ul> </li> <li>● <b>Add</b> <p>The value of the specified header will be returned even if the header does not exist in an API response.</p> </li> </ul> |

## Script Configuration Example

```
{
  "response_headers": [
    {
      "name": "header1",
      "value": "test",
      "action": "append"
    },
    {
      "name": "header2",
      "value": "roma",
      "action": "override"
    }
  ]
}
```

## 7.9.5 Circuit Breaker Plug-in

### Overview

ROMA Connect uses circuit breakers to protect the system when performance issues occur on backend services. When the backend service of an API times out for  $N$  consecutive times or there is long latency, the circuit breaker downgrade mechanism is triggered to return an error to the API caller or forward the request to the specified backend. After the backend service recovers, the circuit breaker closes and the request becomes normal.

### Parameter Description

**Table 7-76** Parameter description

| Parameter            | Description  |
|----------------------|--|
| API Sharing          | When enabled, all APIs bound to the current plug-in are calculated together for circuit breaker triggering.  |
| Circuit Breaker Type | Specify a circuit breaker triggering type. <ul style="list-style-type: none"> <li>• <b>Timeout downgrade:</b> The circuit breaker is triggered upon backend timeout.</li> <li>• <b>Conditional downgrade:</b> The circuit breaker is triggered whenever conditions configured in <b>Match Conditions</b> are met.</li> </ul>   |
| Condition Type       | Specify a circuit breaker triggering mode. <ul style="list-style-type: none"> <li>• <b>Count:</b> Once the number of requests that meet the conditions specified within the time window reaches the threshold, the circuit breaker is immediately triggered.</li> <li>• <b>Percentage:</b> Once the percentage of requests that meet the conditions specified within the time window reaches the threshold, the circuit breaker is triggered at the end of the time window.</li> </ul> |




| Parameter        | Description   |
|------------------|---|
| Match Conditions | <p>This parameter is mandatory only when <b>Circuit Breaker Type</b> is set to <b>Conditional downgrade</b>.</p> <p>Configure the match conditions for circuit breaker triggering.</p> <ul style="list-style-type: none"><li>• <b>Response Error Codes:</b> The circuit breaker is triggered when the backend responds with the status codes specified.</li><li>• <b>Response Latency:</b> The circuit breaker is triggered when the backend response latency exceeds the threshold specified.</li></ul>  |
| Time Window      | <p>Configure the time window for circuit breaker triggering. Use this parameter together with <b>Threshold</b> or <b>Min Percentage</b>. Once the number of times or percentage when conditions are met within the time window exceeds the threshold, the circuit breaker is triggered.</p>   |
| Threshold        | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Count</b>.</p> <p>Configure the threshold for circuit breaker triggering. Use this parameter together with <b>Time Window</b>. Once the number of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered.</p> <p><b>NOTE</b></p> <p>A circuit breaker plug-in is triggered for a single gateway component. If ROMA Connect has multiple components, the triggering for each component is determined separately.</p> <p>If the threshold is reached within the time window for a gateway component, requests sent to the component trigger the circuit breaker, but other gateway components still forward requests normally.</p> <p>An IP address indicates a gateway component. To view the number of gateway components, go to the <b>Instance Information</b> page of a ROMA Connect instance by clicking the instance name on the console and view the number of IP addresses in <b>Connection Addresses</b>.</p> |
| Min Calls        | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Percentage</b>.</p> <p>Configure the minimum number of calls to trigger the circuit breaker. Once the total number of API calls within the time window reaches the threshold specified, the circuit breaker is triggered.</p>   |

| Parameter                | Description  |
|--------------------------|--|
| Min Percentage           | <p>This parameter is mandatory only when <b>Condition Type</b> is set to <b>Percentage</b>.</p> <p>Configure the threshold for circuit breaker triggering. Use this parameter together with <b>Time Window</b>. Once the percentage of backend requests that meet the conditions within the time window reaches the threshold, the circuit breaker is triggered.</p>   |
| Control Duration         | <p>Configure the duration for which the circuit breaker is open. The circuit breaker is closed when the duration reaches the value specified.</p>  |
| Backend Downgrade Policy | <p>Specify whether to enable the backend downgrade policy.</p> <ul style="list-style-type: none"><li>• Enable: Requests for APIs that have triggered a downgrade will be forwarded to a specified backend.</li><li>• Disable: Requests for APIs that have triggered a downgrade will not be forwarded to any backend. Instead, an error message indicating that the service is unavailable will be returned.</li></ul> |

| Parameter           | Description   |
|---------------------|---|
| Backend Policy Type | <p>This parameter is mandatory only when <b>Backend Downgrade Policy</b> is enabled.</p> <p>Specify the backend type to which requests will be forwarded when the circuit breaker is open.</p> <ul style="list-style-type: none"><li>● <b>Mock</b>: The defined response will be returned.<ul style="list-style-type: none"><li>- <b>Status Code</b>: the status code to be included in the response</li><li>- <b>Response</b>: the response body, which is in JSON format</li><li>- <b>Response Header</b>: header parameters to be included in the response</li></ul></li><li>● <b>HTTP&amp;HTTPS</b>: Backend requests will be forwarded to a specified HTTP&amp;HTTPS backend service.<ul style="list-style-type: none"><li>- <b>Use Load Balance Channel</b>: specifies whether to use load balance channels to access the backend. If you want to select <b>yes</b>, <a href="#">create a load balance channel</a> in advance.</li><li>- <b>Backend URL</b>: address of the backend service to forward requests to.</li><li>- <b>Timeout (ms)</b>: backend request timeout. The default value is 5000 ms.</li></ul></li><li>● <b>FunctionGraph</b>: Backend requests will be forwarded to a specified function.<ul style="list-style-type: none"><li>- <b>Function URN</b>: unique identifier of a function request. Click <b>Select Function URN</b> to add a backend function URN.</li><li>- <b>Function Name</b>: automatically displayed after you select a function.</li><li>- <b>Version</b>: version of the function.</li><li>- <b>Invocation Type</b>: invocation type of the function.<br/><b>Synchronous</b>: When receiving an invocation request, FunctionGraph immediately processes the request and returns a result. The client closes the connection once it has received a response from the backend.<br/><b>Asynchronous</b>: After receiving a calling request, FunctionGraph queues the request and returns the result after the request is successfully executed. The server processes the queuing requests one by one when it is idle. The client does not care about the calling result.</li><li>- <b>Timeout (ms)</b>: backend request timeout. The default value is 5000 ms.</li></ul></li><li>● <b>Passthrough</b>: Backend requests will be forwarded to the original API backend.</li></ul> |

| Parameter                    | Description  |
|------------------------------|--|
|                              | <p>To add header parameters to backend requests, click <b>Add Parameter</b>.</p>   |
| Downgrade Parameter Settings | <p>Specify whether to enable downgrade parameter configuration. When enabled, custom rules take precedence over the default ones configured above:</p> <ul style="list-style-type: none"> <li>• If a custom rule is matched, its trigger conditions and downgrade policy take effect. If the custom rule contains no trigger condition or downgrade policy, the default settings in <b>Trigger Configuration</b> and <b>Backend Downgrade Policy</b> take effect.</li> <li>• If the custom rules are not matched, the default settings take effect.</li> </ul> |
| Parameters                   | <p>Define the parameters for rule matching.</p> <ul style="list-style-type: none"> <li>• <b>Parameter Location</b>: position of a parameter in an API request.</li> <li>• <b>Parameter</b>: name of a parameter used for rule matching.</li> </ul> <p>By default, the system contains the <b>reqPath</b> (request path) and <b>method</b> (request method) parameters. Click <b>Add Parameter</b> to add parameters.</p>   |

| Parameter | Description  |
|-----------|--|
| Rules     | <p>Customize matching rules for the circuit breaker. Click <b>Add Rule</b> to add rules. The system matches rules from top to bottom. Adjust the rule priority by moving the rules up or down.</p> <ul style="list-style-type: none"><li>• <b>Conditions:</b> Click  to edit the condition expressions. If there are three or more expressions, you can layer them by clicking <b>Set Lower Level</b>.<ul style="list-style-type: none"><li>- =: equal</li><li>- !=: not equal to</li><li>- <b>pattern:</b> regular expression</li><li>- <b>enum:</b> enumerated values. Separate them with commas (,).</li></ul></li><li>• For details about how to configure the triggering conditions and backend downgrade, see the instructions for the default settings above.</li></ul> <p>Example: You have enabled <b>Downgrade Parameter Settings</b> and added rules <b>rule01</b> and <b>rule02</b> in sequence. And you have disabled <b>Trigger Configuration</b> and enabled <b>Backend Downgrade Policy</b> for <b>rule01</b>, and have enabled both parameters for <b>rule02</b>. Analysis: The circuit breaker first checks whether the match conditions of <b>rule01</b> are met. If yes, the circuit breaker opens based on the default settings because no trigger condition is defined in <b>rule01</b>, and the backend downgrade policy in <b>rule01</b> is executed. If no, <b>rule02</b> is checked.</p> |

## Script Configuration Example

```
{
  "breaker_condition":{
    "breaker_type":"timeout",
    "breaker_mode":"counter",
    "unhealthy_threshold":30,
    "time_window":15,
    "open_breaker_time":15,
    "unhealthy_percentage":51,
    "min_call_threshold":20
  },
  "scope":"share",
  "downgrade_default":{
    "type":"http",
    "passthrough_infos":null,
    "func_info":null,
    "mock_info":null,
    "http_info":{
      "isVpc":false,
      "vpc_channel_id":"",
      "address":"10.10.10.10",
      "scheme":"HTTP",
      "method":"GET",
      "path":"/demo",
      "timeout":5000
    }
  }
}
```

```
    },
    "http_vpc_info":null
  },
  "downgrade_parameters":[
  {
    "name":"reqPath",
    "type":"path",
    "value":"path",
    "disabled":true,
    "focused":true,
    "id":"92002eqbpilg6g"
  },
  {
    "name":"method",
    "type":"method",
    "value":"method",
    "disabled":true,
    "focused":true,
    "id":"tuvxetsdqvcos8"
  }
  ],
  "downgrade_rules":[
  {
    "rule_name":"rule-test1",
    "parameters":[
      "reqPath",
      "method"
    ],
    "match_regex":"[\\\"reqPath\\\",\\\"==\\\",\\\"/test\\\"]",
    "downgrade_backend":{
      "type":"mock",
      "passthrough_infos":null,
      "func_info":null,
      "mock_info":{
        "status_code":200,
        "result_content":"{status: ok}",
        "headers":[]
      },
      "http_info":null,
      "http_vpc_info":null
    },
    "breaker_condition":{
      "breaker_type":"timeout",
      "breaker_mode":"percentage",
      "unhealthy_threshold":30,
      "time_window":15,
      "open_breaker_time":15,
      "unhealthy_percentage":51,
      "min_call_threshold":20
    }
  }
  ]
}
```

## 7.10 Configuring a Custom Authorizer

### 7.10.1 Creating a Frontend Custom Authorizer

#### Overview

To use your own API calling authentication system, create a custom authorizer.

Custom authorizers are classified into the following types:

- Frontend: ROMA Connect uses a custom authentication function to authenticate API requests.

- Backend: The backend service of an API uses a custom authentication function to authenticate requests forwarded by ROMA Connect.

This section describes how to create a frontend custom authorizer. Ensure that you have created a function backend before using it for a custom authorizer.

## Creating a Function Backend for Frontend Authentication

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Create Backend**.
3. On the **Create Backend** page, set backend parameters and click **Create**.
  - **Backend Request Method** must be set to **POST**.
  - The input parameters are not required. The function backend will obtain the values of the header and query parameters from API requests.
  - For details about the settings of other parameters, see [Creating a Function API](#).

After the backend is created, the online IDE page is automatically displayed.

4. Develop a function backend.

In the upper left corner of the online IDE, choose **File > Create Function Backend > Blank Template**. In the dialog box displayed, click **Yes**. Compile a function script for security authentication and click **Save**.

The function script used for frontend custom authentication must meet the following conditions:

- For the request parameters obtained using the function script:
  - Header parameter: indicates the identity source parameter in Header defined in custom authentication. The parameter value is transferred from the API request that uses the frontend custom authentication. The called parameter in the function script is in the format of: **body["headers"]["Parameter name"]**.
  - Query parameter: indicates the identity source parameter in Query defined in custom authentication. The parameter value is transferred from the API request that uses the frontend custom authentication. The called parameter in the function script is in the format of: **body["queryStringParameters"]["Parameter name"]**.
  - Body parameter: user data defined when a custom authorizer is created. The format of calling the body parameter is **body["user\_data"]**.

### NOTE

During frontend custom authentication, the header and query parameters of API requests are placed in the **headers** and **queryStringParameters** parameters of the backend request body and transferred to the authentication function. Therefore, when the header and query parameters need to be called in a function script, the parameters need to be obtained from the backend request body. For details about the examples of **headers** and **queryStringParameters** in the backend request body, see [Test Procedure Examples](#).

### – Response

The response body cannot be greater than 1 MB. The response content must meet the following format:

```
{
  "status": "allow/deny",
  "context": {
    "user": "abc"
  }
}
```

- **status:** identifies the authentication result. This field is mandatory. Only **allow** or **deny** is supported. **allow** indicates that the authentication is successful, and **deny** indicates that the authentication fails.

- **context:** indicates the authentication response result. This field is mandatory. Only key-value pairs of the string type are supported. The key value does not support JSON objects or arrays.

The data in the context is user-defined. After the authentication is successful, the data can be used as a **system parameter** (frontend authentication parameter) and mapped to the backend request parameter of the API. The system parameter name set in the API backend service must be the same as the parameter name in the context. The parameter name is case sensitive. The parameter name in **context** must start with a letter and contain 1 to 32 characters, including letters, digits, underscores (\_), and hyphens (-).

The following is an example of the Header parameter definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["headers"]["test"]=="abc"){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

The following is an example of the Query parameter definition script:

```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["queryStringParameters"]["test"]=="abc"){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

The following is an example of the Body parameter definition script:



```
function execute(data){
  data=JSON.parse(data)
  body=data.body
  if(body["user_data"]=='abc'){
    return{
      "status": "allow",
      "context": {
        "user": "abcd"
      }
    }
  }else{
    return{
      "status": "deny"
    }
  }
}
```

5. Test the function backend.

In the upper right corner of the page, click **Test**. In the **Test Parameters** area, add request parameters required for authentication based on the definition of the function backend and click **Test** to send the request. If the value of status in the test result is **allow**, the test is successful.

In the **Test Parameters** area, set backend request parameters. The Header, Query, and Body authentication parameters must be set in the Body parameter of the backend request. Based on the preceding script examples, the following describes how to set the authentication parameters:

- Header parameter

```
{
  "headers":{
    "test":"abc"
  }
}
```

- Query parameter

```
{
  "queryStringParameters":{
    "test":"abc"
  }
}
```

- Body parameter

```
{
  "user_data": "abc"
}
```

6. Deploy the function backend.

After the backend test is complete, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the function backend.

## Creating a Frontend Custom Authorizer

Before creating a frontend custom authorizer, ensure that the function backend used for frontend custom authentication has been created. Otherwise, create a function backend first. For details, see [Creating a Function Backend for Frontend Authentication](#).

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Custom Authorizers** tab page, click **Create**.

3. In the **Create Custom Authorizer** dialog box, configure custom authorizer information and click **Create**.

**Table 7-77** Parameters for creating a frontend custom authorizer

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a custom authorizer name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Type                    | Select <b>Frontend</b> .  |
| Integration Application | Select an integration application for the custom authorizer.  |
| Function URN            | Select a function backend for frontend custom authentication. Only function backends in the <b>Deployed</b> state can be selected.  |
| Identity Sources        | Add header or query request parameters used for authentication.<br>If the value of <b>Cache Duration</b> is not <b>0</b> , a request parameter must be added. When the authentication result is cached, this parameter is used as the cache index of the authentication result. |
| Max. Cache Age (s)      | Set the time for caching authentication results. Value <b>0</b> means that authentication results will not be cached. The maximum value is <b>3600</b> .  |
| Send Request Body       | Determine whether to send the API request body to the authentication function.  |
| User Data               | Enter custom parameters, which will be used together with <b>Identity Sources</b> for request authentication.   |

## 7.10.2 Creating a Backend Custom Authorizer

### Overview

To use your own backend request authentication system, create a custom authorizer.

Custom authorizers are classified into the following types:

- **Frontend:** ROMA Connect uses a custom authentication function to authenticate API requests.
- **Backend:** The backend service of an API uses a custom authentication function to authenticate requests forwarded by ROMA Connect.

This section describes how to create a backend custom authorizer. Ensure that you have created a function backend before using it for a custom authorizer.

## Creating a Function Backend for Backend Authentication

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > Custom Backends**. On the **Backends** tab page, click **Create Backend**.
3. On the **Create Backend** page, set backend parameters and click **Create**.
  - **Backend Request Method** must be set to **POST**.
  - The input parameters are not required. The header and query parameters are not used for the backend custom authorization.
  - For details about the settings of other parameters, see [Creating a Function API](#).

After the backend is created, the online IDE page is automatically displayed.

4. Develop a function backend.

In the upper left corner of the online IDE, choose **File > Create Function Backend > Blank Template**. In the dialog box displayed, click **Yes**. Compile a function script for security authentication and click **Save**.

The function script used for backend custom authentication must meet the following conditions:

- Request parameter  
Body parameter: user data defined when a custom authorizer is created. The format of calling the body parameter is `body["user_data"]`.
- Response  
The response body cannot be greater than 1 MB. The response content must meet the following format:

```
{
  "status": "allow/deny",
  "context": {
    "user": "abc"
  }
}
```

- **status**: identifies the authentication result. This field is mandatory. Only **allow** or **deny** is supported. **allow** indicates that the authentication is successful, and **deny** indicates that the authentication fails.
- **context**: indicates the authentication response result. This field is mandatory. Only key-value pairs of the string type are supported. The key value does not support JSON objects or arrays.  
The data in the context is user-defined. After the authentication is successful, the data can be used as a **system parameter** (backend authentication parameter) and mapped to the backend request parameter of the API. The system parameter name set in the API backend service must be the same as the parameter name in the context. The parameter name is case sensitive. The parameter name in **context** must start with a letter and contain 1 to 32 characters, including letters, digits, underscores (\_), and hyphens (-).

The following is an example of the user data definition script:

```
function execute(data){
  data=JSON.parse(data)
```

```
body=data.body
if(body["user_data"]=='abc'){
  return{
    "status": "allow",
    "context": {
      "user": "abcd"
    }
  }
}else{
  return{
    "status": "deny"
  }
}
```

5. Test the function backend.

In the upper right corner of the page, click **Test**. In the **Test Parameters** area, add request parameters required for authentication based on the definition of the function backend and click **Test** to send the request.

The user data definition script in the preceding step is used as an example. Enter the request content `{"user_data": "abc"}` in the body parameter to authenticate backend service requests.

If the value of status in the test result is **allow**, the test is successful.

6. Deploy the function backend.

After the backend test is complete, click **Deploy** in the upper right corner of the page. In the dialog box displayed, click **Yes** to deploy the function backend.

## Creating a Backend Custom Authorizer

Before creating a backend custom authorizer, ensure that the function backend used for backend custom authentication has been created. Otherwise, create a function API first. For details, see [Creating a Function Backend for Backend Authentication](#).

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Custom Authorizers** tab page, click **Create**.
3. In the **Create Custom Authorizer** dialog box, configure custom authorizer information and click **Create**.

**Table 7-78** Parameters for creating a backend custom authorizer

| Parameter               | Description   |
|-------------------------|---|
| Name                    | Enter a custom authorizer name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Type                    | Select <b>Backend</b> .   |
| Integration Application | Select an integration application for the custom authorizer.  |

| Parameter          | Description  |
|--------------------|--|
| Function URN       | Select a function backend for custom authentication. Only function backends in the <b>Deployed</b> state can be selected.                                |
| Max. Cache Age (s) | Set the time for caching authentication results. Value <b>0</b> means that authentication results will not be cached. The maximum value is <b>3600</b> . |
| Send Request Body  | Specify whether to send the request body to the function.  |
| User Data          | Enter the parameters for creating a custom authorizer.   |

## 7.11 Configuring Signature Verification for Backend Services

### Overview

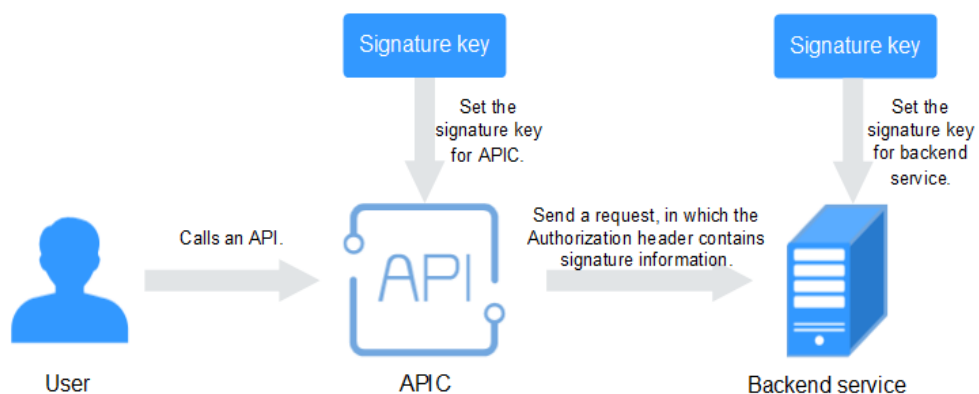
Signature keys are used by backend services to verify the identity of ROMA Connect.

A signature key consists of a key and a secret. The signature key takes effect only after it is bound to an API.

#### NOTE

An API can be bound to only one signature key in an environment, but a signature key can be bound to multiple APIs.

After a signature key is bound to an API, ROMA Connect uses the key and secret in the signature key to add signature information to requests sent to the backend service of the API. The backend service needs to sign the requests in the same way. If the signature is the same as that included in the **Authorization** header of the requests, the backend service determines that the requests sent by ROMA Connect are valid.



## Creating a Signature Key

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Signature Keys** tab page, click **Create**.
3. In the **Create Signature Key** dialog box, configure signature key information.

**Table 7-79** Parameters for creating a signature key

| Parameter           | Description   |
|---------------------|---|
| Name                | Enter a signature key name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Type                | Select the type of the signature key. The options are <b>hmac</b> , <b>aes</b> , and <b>basic</b> .   |
| Key                 | Set the key based on the signature key type you have selected. <ul style="list-style-type: none"><li>• If <b>Type</b> is set to <b>hmac</b>, enter the key in the key pair used for HMAC authentication.</li><li>• If <b>Type</b> is set to <b>basic</b>, enter the username used for basic authentication.</li><li>• If <b>Type</b> is set to <b>aes</b>, enter the key used for AES authentication.</li></ul>       |
| Signature Algorithm | Select an AES signature algorithm. Options: <ul style="list-style-type: none"><li>• aes-128-cfb</li><li>• aes-256-cfb</li></ul>   |
| Secret              | Set the key based on the signature key type you have selected. <ul style="list-style-type: none"><li>• If <b>Type</b> is set to <b>hmac</b>, enter the secret in the key pair used for HMAC authentication.</li><li>• If <b>Type</b> is set to <b>basic</b>, enter the password used for basic authentication.</li><li>• If <b>Type</b> is set to <b>aes</b>, enter the vector used for AES authentication.</li></ul> |
| Confirm Secret      | Enter the same secret again.  |

4. Click **OK**.  
After the signature key is created, you also need to perform the operations described in [Binding a Signature Key to an API](#) to make the signature key take effect for the API.

## Binding a Signature Key to an API

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.

2. In the navigation pane on the left, choose **API Connect > API Management**. On the **Signature Keys** tab page, click **Bind to API**.
3. On the **Bind to API** page, click **Select API**.
4. In the **Select API** dialog box, select the APIs to which the signature key is to be bound in the specified environment.  
APIs can be filtered by API group, environment, and API name.
5. Click **OK**.

## Configuring Signature Verification for Backend Services

After binding a signature key to APIs, develop signature verification for backend services to verify request signatures. For details, see [Developing Signature Verification for Backend Services](#).

# 7.12 Configuring API Cascading

## Overview

API cascading allows you to cascade two ROMA Connect instances in the same region or different regions so that one instance can use an API of the other instance as its backend service, thereby implementing cross-instance API calling. A dedicated authentication channel will be used for API cascading to prevent authentication conflict.

- Cascading instance: uses an API of the other instance as its backend service.
- Cascaded instance: provides its API to the other instance as a backend service.


API cascading enables you to provide APIs in one instance for another instance to improve the reuse capability of API assets, without having to deploy backend services in different instances repeatedly.

## Prerequisites

- The two instances can communicate with each other.
- If the two instances are located in different networks and communicate with each other through an air wall, their IP address and port number must be configured on the air wall. In addition, the TCP protocol must be used on the air wall for secure access. A dedicated VPN or tunnel can also be used to implement cross-network interworking.


## Procedure

1. Enable cascading for the cascaded instance.
  - a. Log in to the ROMA Connect console on which the cascaded instance is located. On the **Instances** page, click **View Console**.
  - b. On the **Instance Information** page, click the **Configuration Parameters** tab and locate the **cascade** parameter.
  - c. Click **Edit** on the right of the parameter, set **Current Value** to **on**, and click **Save**.

- d. Click  on the left of the parameter and configure the following parameters.

**Table 7-80** Parameters related to the cascading function

| Parameter            | Description  |
|----------------------|--|
| cascade_auth_key     | Encryption key used for authentication between two APIs. The <b>cascade_auth_key</b> value of the cascaded instance must be the same as that of the cascading instance.    |
| cascade_instance_ids | IDs of the instances that are allowed to cascade the current instance. Use commas (,) to separate multiple instance IDs. A maximum of five instance IDs can be configured. |

2. Enable cascading for the cascading instance.
  - a. Log in to the ROMA Connect console on which the cascading instance is located. On the **Instances** page, click **View Console**.
  - b. On the **Instance Information** page, click the **Configuration Parameters** tab and locate the **cascade** parameter.
  - c. Click **Edit** on the right of the parameter, set **Current Value** to **on**, and click **Save**.
  - d. Click  on the left of the parameter and configure the following parameters.

**Table 7-81** Parameters related to the cascading function

| Parameter            | Description   |
|----------------------|---|
| cascade_auth_key     | Encryption key used for authentication between two APIs. The <b>cascade_auth_key</b> value of the cascaded instance must be the same as that of the cascading instance. |
| cascade_instance_ids | This parameter is not required for cascading instances.   |

3. Create a load balance channel from the cascading instance to the cascaded instance.
  - a. In the navigation pane of the cascading instance console, choose **API Connect > API Management**. On the **Load Balance Channels** tab page, click **Create**.
  - b. On the **Create Load Balance Channel** page, configure load balance channel information.
    - Configure basic information about the load balance channel.



**Table 7-82** Parameters for configuring the load balance channel

| Parameter           | Description  |
|---------------------|--|
| Name                | Enter a load balance channel name. It is recommended that you enter a name based on naming rules to facilitate search.   |
| Port                | Enter the access port number of the ECS in the load balance channel. You can determine the port number based on the protocol used by the API in the cascaded instance. For HTTP, set this parameter to <b>80</b> . For HTTPS, set this parameter to <b>443</b> . |
| Routing Algorithm   | Select an algorithm for routing backend service requests. The load balance channel determines the server to which the requests are to be sent by the algorithm.  |
| Backend Server Type | Select the type of servers for the load balance channel. To access the API of the cascaded instance, select <b>Cloud server</b> .  |

- Configure the backend server type.
  - 1) Click **Create Server Group**.
  - 2) In the dialog box displayed, configure group information and click **OK**.

Servers can be added to different groups.

**Table 7-83** Server group configuration

| Parameter   | Description  |
|-------------|--|
| Group Name  | Enter a server group name. It is recommended that you enter a name based on naming rules to facilitate search.                   |
| Weight      | Enter the weight of the server group. A larger weight indicates that more requests can be forwarded to the servers in the group. |
| Description | Enter a brief description of the server group.   |

- 3) Click **Add Backend Server Address**.
- 4) Configure backend server information.

**Table 7-84** Backend server information

| Parameter              | Description   |
|------------------------|---|
| Backend Server Address | Enter the API access address of the cascaded instance. <ul style="list-style-type: none"><li>• IP address format:<ul style="list-style-type: none"><li>– Set this parameter to the EIP of the cascaded instance if the two instances communicate with each other over a public network.</li><li>– Set this parameter to the APIC connection address of the cascaded instance if the two instances communicate with each other over a VPC intranet.</li></ul></li><li>• Domain name format: Enter the access domain name of the API.</li></ul> |
| Standby Node           | After you enable this option, the backend server serves as a standby node. It works only when all non-standby nodes are faulty.   |
| Port                   | Enter the access port number of the backend server. If the port number is <b>0</b> , the port of the load balance channel is used.  |
| Server Status          | Specify whether to enable the server. Requests are distributed to the server only after it is enabled.  |

- Configure the health check.

The health check function is enabled by default. If you do not need to perform the health check, disable this function.

**Table 7-85** Health check configurations

| Parameter              | Description  |
|------------------------|--|
| Protocol               | Select the protocol used for the health check. The value can be <b>TCP</b> , <b>HTTP</b> , or <b>HTTPS</b> .   |
| Two-way Authentication | This parameter is available only if <b>Protocol</b> is set to <b>HTTPS</b> .<br>Specify whether to enable two-way authentication between ROMA Connect and backend servers. |
| Path                   | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Enter the health check URL.  |

| Parameter           | Description  |
|---------------------|--|
| Method              | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>Select the HTTP request method used for the health check. The value can be <b>GET</b> or <b>HEAD</b> .   |
| Health Check Port   | Enter the destination port of the health check. By default, the port number configured for the load balance channel is used.   |
| Healthy Threshold   | Number of consecutive successful checks required for an ECS to be considered healthy. For example, if <b>Healthy Threshold</b> is set to <b>2</b> , ROMA Connect considers the ECS status as healthy when the check is successful for two consecutive times. |
| Unhealthy Threshold | Number of consecutive failed checks required for an ECS to be considered unhealthy. For example, if <b>Unhealthy Threshold</b> is set to <b>5</b> , ROMA Connect considers the ECS status as abnormal when the check fails for five consecutive times.       |
| Timeout (s)         | Response timeout of a health check, in seconds. If no response is received within the specified duration, the health check fails.  |
| Interval (s)        | Interval between consecutive checks.   |
| Response Codes      | Mandatory for <b>Protocol</b> set to <b>HTTP</b> or <b>HTTPS</b> .<br>When the server returns a specified HTTP response code, the server considers the response as successful. Multiple response codes can be specified at the same time.                    |

- c. Click **Finish**.
4. On the cascading instance, create an API and set the backend address to the API in the cascaded instance.

For details about how to create an API, see [Creating an API](#). Only the configuration of defining the backend service is different between the cascading and the cascaded instances, as shown as follows.

**Table 7-86** Backend service access parameters

| Parameter    | Description   |
|--------------|---|
| Backend Type | Select a backend service type. When the API of the cascaded instance is used as the backend service, select <b>HTTP/HTTPS</b> . |
| Protocol     | Set this parameter based on the request protocol of the API in the cascaded instance.   |

| Parameter              | Description  |
|------------------------|--|
| Request Mode           | Set this parameter based on the API request method used in the cascaded instance.  |
| Load Balancing         | Determine whether to use a load balance channel to access backend services. When the API of the cascaded instance is used as the backend service, select <b>Configure now</b> .  |
| Load Balance Channel   | Select the load balance channel created in <a href="#">3</a> .   |
| Cascading Flag         | This parameter is available only when <b>cascade</b> in the <a href="#">instance configuration parameters</a> is set to <b>on</b> .<br>Determine whether to use the cascading mode to access backend services. Enable this option.   |
| Host Header            | Define the Host header field carried in the backend service request. If you have specified <b>Backend Server Address</b> with an IP address when creating a load balance channel in <a href="#">3</a> , set <b>Host Header</b> to the domain name of the API of the cascaded instance.   |
| Path                   | Enter the request path of the backend service in the <code>/getUserInfo/{userId}</code> format. A URL can have multiple path parameters, each enclosed by braces.<br>If the path needs to contain an environment variable, enclose the environment variable in number signs (#), for example, <code>/#path#</code> . Multiple environment variables can be added, for example, <code>/#path##request#</code> .   |
| Timeout (ms)           | Enter the timeout interval of a backend service request. The default value is <b>5000</b> .  |
| Retry No.              | Number of retry times after ROMA Connect fails to call the backend service. <ul style="list-style-type: none"><li>• If the value is <b>-1</b>, the retry function is disabled. However, requests will be retried once by default except for those using <b>POST</b> and <b>PATCH</b>.</li><li>• If the value ranges from 0 to 10, the retry function is enabled, and retries are performed based on the configured value. The number of retries must be less than the number of backend servers enabled in the load balance channel.</li></ul> |
| Two-way Authentication | This parameter is available only if <b>Protocol</b> is set to <b>HTTPS</b> .<br>Determine whether to enable two-way authentication between ROMA Connect and backend services. When the API of the cascaded instance is used as the backend service, do not enable two-way authentication.  |

| Parameter              | Description   |
|------------------------|---|
| Backend Authentication | Determine whether to enable backend authentication. When the API of the cascaded instance is used as the backend service, do not enable backend authentication. |

# 8 Message Integration Guide

---

- [Usage Introduction](#)
- [Creating a Topic](#)
- [\(Optional\) Granting Permissions for Topics](#)
- [Connecting to a Topic](#)
- [Topic Management](#)
- [Migrating Kafka Services](#)

## 8.1 Usage Introduction

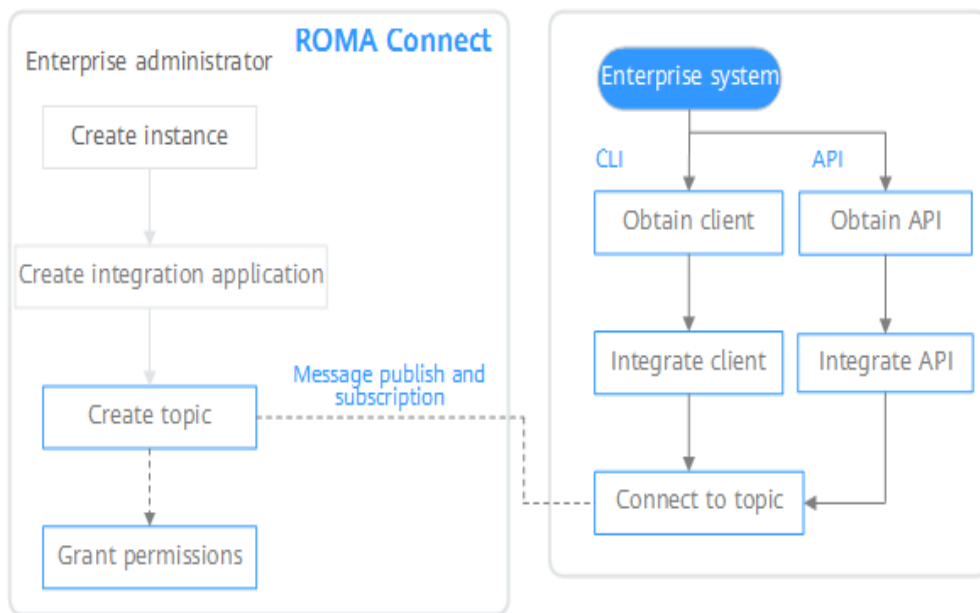
### Function Description

MQS is a message integration component of ROMA Connect. It uses a unified message access mechanism to provide secure and standard message channels for cross-network access. ROMA Connect has the following advantages for message integration:

- **Support for Kafka-native features**  
DMS for Kafka is compatible with open-source Kafka APIs and supports all message processing functions of open-source Kafka.
- **Secure message transmission**  
Operations on RabbitMQ instances are recorded and can be audited. Messages can be encrypted before storage. SASL authentication and security groups are used to enhance network access control.
- **High reliability of message data**  
MQS instances support data persistence and replication. Messages can be replicated synchronously or asynchronously between replicas.

### Process Flow

The following figure shows the process of using ROMA Connect for message integration.

**Figure 8-1** Process of using ROMA Connect for message integration

1. You have **created an instance and integration application**.
2. **Create a topic.**  
Create a topic for storing messages so that message producers can publish messages and message consumers can subscribe to messages.
3. (Optional) **Grant permissions for topics.**  
If SASL\_SSL of MQS is enabled in the ROMA Connect instance, the client needs to use the key and secret of the authorized integration application for security authentication when publishing and subscribing to messages to the topic.
4. **Connect to the topic.**  
The open-source Kafka client needs to be integrated to the system. Then, messages can be produced and consumed through the command lines provided by the client.

## 8.2 Creating a Topic

### Overview

Create a topic for storing messages so that producers can publish messages and consumers can subscribe to messages in it.

### Prerequisites

Each topic must belong to an integration application. Before creating a topic, ensure that an integration application is available. Otherwise, **create an integration application** first.

## Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane, choose **Message Queue Service > Topic Management**. On the page displayed, click **Create Topic** in the upper right corner.
3. In the dialog box displayed, configure topic information and click **OK**.

**Table 8-1** Parameters for creating a topic

| Parameter               | Description  |
|-------------------------|--|
| Topic Name              | Enter a topic name. Using naming rules facilitates future search.<br><b>NOTE</b><br>Periods (.) and underscores (_) cannot be used to distinguish names. For example, if you create a topic named <b>Topic.test</b> first, you cannot create another topic named <b>Topic_test</b> .   |
| Integration Application | Select the integration application to which the topic belongs.   |
| Permission              | Select the permission on a topic for the integration application to which the topic belongs. <ul style="list-style-type: none"><li>● <b>Publish/Subscribe</b>: Messages can be published to and subscribed from the topic.</li><li>● <b>Publish</b>: Messages can only be published to the topic.</li><li>● <b>Subscribe</b>: Messages can be subscribed only in the topic.</li></ul>                        |
| Partitions              | Set the number of partitions of a topic to improve the concurrent performance of message production and consumption.<br>If this parameter is set to <b>1</b> , messages will be consumed in the FIFO order.  |
| Replicas                | Set the number of replicas of a topic. ROMA Connect automatically backs up data on each replica. If one replica is faulty, data will still be available. The reliability increases with the number of replicas of a topic.<br><b>NOTE</b><br>If a broker is faulty, an internal error will occur when you query messages using a topic with only one replica. Use topics with more than one replica instead. |
| Aging Time (h)          | Set the aging time of messages stored in the topic. Messages that exceed the aging time will be deleted.   |



| Parameter               | Description   |
|-------------------------|---|
| Synchronous Replication | When a client creates a message to a topic, ROMA Connect determines whether to replicate the message to all replicas and then returns a response to the client.<br>After synchronous replication is enabled, you also need to set <b>acks</b> to <b>-1</b> on the client. Otherwise, synchronous replication will not take effect.  |
| Synchronous Flushing    | This parameter specifies whether each message created by a client to the topic is immediately written to the disk. When synchronous flushing is enabled, the reliability is enhanced.   |
| Tag Name                | Add message publishing and subscription tags for the integration application to which a topic belongs. Generally, you do not need to add the tags. The tags are used only in special service scenarios.<br>After a tag is added, the client needs to transmit the tag information when publishing and subscribing to messages in the topic. If there are multiple tags, the tags transmitted by the client must be a subset of the tags contained in the topic. |
| Filter                  | Enter values to filter out topic messages containing these values.  |
| Description             | Enter a brief description of the topic.   |

## 8.3 (Optional) Granting Permissions for Topics

### Overview

You need to grant permissions to integration applications only if MQS SASL\_SSL is enabled for the ROMA Connect instance so that the integration applications can send messages to or receive messages from topics. When a client publishes messages into a topic and subscribes to a topic to obtain messages, the key and secret of the authorized integration application must be used for security authentication.

By default, the integration application to which a topic belongs has the permissions to publish and subscribe to messages in the topic.

### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Message Queue Service > Topic Management**. On the page displayed, click **Grant Integration Application Permission** in the **Operation** column of the topic to which you want to grant permissions.

3. In the **Grant Integration Application Permission** dialog box, grant permissions for integration applications.  
In the **Available** area, select the integration applications to be authorized. In the **Selected/All** area, configure the topic permissions for the selected integration applications.

**Table 8-2** Authorization configuration

| Parameter  | Description  |
|------------|--|
| Permission | Select the topic permissions of the integration application. <ul style="list-style-type: none"><li>● <b>Publish/Subscribe:</b> Messages can be published to and subscribed from a topic.</li><li>● <b>Publish:</b> Messages can be only published to a topic.</li><li>● <b>Subscribe:</b> Messages can be only subscribed from a topic.</li></ul>  |
| Tag        | Add tags for publishing and subscribing to messages of a topic. Tags are used only in special service scenarios and do not need to be configured in general service scenarios.<br><br>After a tag is added, the client needs to transmit the tag information when publishing and subscribing to messages in the topic. If there are multiple tags, the tags transmitted by the client must be a subset of the tags contained in the topic. |

4. Click **OK**.

## 8.4 Connecting to a Topic

### Overview

After a topic is created, you can use the CLI or open-source Kafka client to connect to the topic and produce and consume messages in the topic.

This section describes how to connect to a topic using CLI. If you use an open-source Kafka client, see [Connecting to MQS in Client Mode](#).

### Prerequisites

- A topic is available. If no topic is available, [create a topic](#) first.
- You have downloaded the open-source Kafka CLI tool whose version matches the Kafka version of the ROMA Connect instance.

You can view the Kafka version in the **MQS Information** area on the **Instance Information** page of the ROMA Connect console.

- [Kafka command line tool 1.1.0](#)
- [Kafka command line tool 2.7.2](#)

- **Java JDK** has been installed in the environment where the Kafka command-line tool is used, and related environment variables have been configured.
- If **MQS SASL\_SSL** has been enabled for the ROMA Connect instance, log in to the ROMA Connect console, choose **Message Queue Service > Topic Management**, and click **Download SSL Certificate** to download the **client certificate**.
- If both SASL\_SSL and intra-VPC plaintext access are enabled for MQS of the ROMA Connect instance, the SASL mode cannot be used for connecting to MQS topics in the VPC.
- If the SASL mode is used for connecting to MQS topics, you are advised to configure the mapping between the host and IP address in the **/etc/hosts** file on the host where the client is located. Otherwise, network delay will occur. Set the IP address to the connection address of MQS. Set the host to the name of each instance host. Ensure that the name of each host is unique. For example:  
10.10.10.11 host01  
10.10.10.12 host02  
10.10.10.13 host03
- A maximum of 500 consumers in the same consumer group can connect to the same MQS. If the number of consumers exceeds 500, the connection fails. If a consumer group with over 500 consumers needs to connect to an MQS, put the consumers into multiple consumer groups.

## Using SASL Authentication

If SASL\_SSL access is enabled for the ROMA Connect instance, messages produced and consumed by the client to the topic are encrypted for transmission, which is more secure. The following uses Linux as an example.

1. Decompress the Kafka command-line tool and client certificates.

Access the directory where the tool package is stored and run the following command to decompress the package:

- Run the following command to decompress the command-line tool package:

```
tar -zxf kafka_tar
```

In the preceding command, **kafka\_tar** indicates the name of the Kafka CLI package.

- Run the following command to decompress the client certificate file package:

```
unzip cert_zip
```

In the preceding command, **cert\_zip** indicates the name of the client certificate file package.

2. Modify the configuration file of the Kafka command-line tool.

Find the **consumer.properties** and **producer.properties** files in the **/config** directory of the Kafka command-line tool and add the following content to the files:

```
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \  
username="*****" \  
password="*****",  
sasl.mechanism=PLAIN
```

```
security.protocol=SASL_SSL
ssl.truststore.location=/cert/client.truststore.jks
ssl.truststore.password=dms@kafka
ssl.endpoint.identification.algorithm=
```

- The values of **username** and **password** are the key and secret of the integration application to which the topic belongs. For details on how to obtain the key and secret, see [Viewing and Editing an Integration Application](#).
  - The value of **ssl.truststore.location** is the path for storing the client certificate obtained after decompression in **1**. Replace it with the actual path. Note that the certificate path in Windows must contain slashes (/).
  - **ssl.truststore.password** indicates the server certificate password, which must be set to **dms@kafka** and cannot be changed.
3. Access the **/bin** directory of the Kafka command-line tool.  
In the Windows operating system, you need to access the **/bin/windows** directory.
  4. Produce a message to the topic.
    - a. Run the following command to create a connection with the topic for producing messages:

```
./kafka-console-producer.sh --broker-list Address --topic TopicName --producer.config ../config/producer.properties
```

In the preceding command:
      - **Address** indicates the MQS address of ROMA Connect. For details about how to obtain the MQS address, see [Viewing Details of an Instance](#). If you access the topic from a public network, use the public connection address. If you access the topic using the intra-VPC mode, use the intranet connection address.
      - **TopicName** indicates the name of the topic where the message is to be produced.
      - **../config/producer.properties** indicates the relative path of the configuration file.
    - b. Enter the message contents and send the messages to the topic.

```
>Message1
>Message2
>Message3
```

In the preceding information, **Message1**, **Message2**, and **Message3** indicate the actual message contents sent to the topic. Each row indicates a message.
    - c. To disconnect from the topic, press **Ctrl+C**.
  5. Consume messages from a topic.

#### NOTE

When a consumer consumes messages from multiple partitions of a topic, messages in only one partition can be consumed at a time. Messages in multiple partitions are consumed in multiple times.

- a. Run the following command to create a connection with the topic for reading messages:

```
./kafka-console-consumer.sh --bootstrap-server Address --topic TopicName --from-beginning --consumer.config ../config/consumer.properties
```

In the preceding command:

- **Address** indicates the MQS address of ROMA Connect. For details about how to obtain the MQS address, see [Viewing Details of an Instance](#). If you access the topic from a public network, use the public connection address. If you access the topic using the intra-VPC mode, use the intranet connection address.
  - **TopicName** indicates the name of the topic where the message is to be consumed.
  - **../config/consumer.properties** indicates the relative path of the configuration file.
- b. After the command is executed, the system is continuously connected to the topic and reads messages. To disconnect from the topic, press **Ctrl+C**.

## Without Using SASL Authentication

If SASL\_SSL access is not enabled for the ROMA Connect instance, the client does not need to load the certificate. Messages produced and consumed by the topic are not encrypted. The following uses Linux as an example.

1. Decompress the Kafka command-line tool.

Access the directory where the tool package is stored and run the following command to decompress the package:

```
tar -zxvf kafka_tar
```

In the preceding command, **kafka\_tar** indicates the name of the Kafka CLI package.

2. Access the **/bin** directory of the Kafka command-line tool.

In the Windows operating system, you need to access the **/bin/windows** directory.

3. Produce a message to the topic.

- a. Run the following command to create a connection with the topic for producing messages:

```
./kafka-console-producer.sh --broker-list Address --topic TopicName
```

In the preceding command:

- **Address** indicates the MQS address of ROMA Connect. For details about how to obtain the MQS address, see [Viewing Details of an Instance](#). If you access the topic from a public network, use the public connection address. If you access the topic using the intra-VPC mode, use the intranet connection address.
  - **TopicName** indicates the name of the topic where the message is to be produced.
- b. Enter the message contents and send the messages to the topic.
- ```
>Message1  
>Message2  
>Message3
```

In the preceding information, **Message1**, **Message2**, and **Message3** indicate the actual message contents sent to the topic. Each row indicates a message.

- c. To disconnect from the topic, press **Ctrl+C**.
4. Consume messages from a topic.

#### NOTE

When a consumer consumes messages from multiple partitions of a topic, messages in only one partition can be consumed at a time. Messages in multiple partitions are consumed in multiple times.

- a. Run the following command to create a connection with the topic for reading messages:

```
./kafka-console-consumer.sh --bootstrap-server Address --topic TopicName --from-beginning
```

In the preceding command:

- **Address** indicates the MQS address of ROMA Connect. For details about how to obtain the MQS address, see [Viewing Details of an Instance](#). If you access the topic from a public network, use the public connection address. If you access the topic using the intra-VPC mode, use the intranet connection address.
  - **TopicName** indicates the name of the topic where the message is to be produced.
- b. After the command is executed, the system is continuously connected to the topic and reads messages. To disconnect from the topic, press **Ctrl+C**.

## 8.5 Topic Management

### 8.5.1 Viewing Message Body

#### Overview

ROMA Connect provides a visualized message query function, which allows you to view the message data stored in topics on the console and view the message body more intuitively and conveniently.

Messages of only one topic can be queried at a time.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Message Queue Service > Message Query**.
3. In the upper right corner of the page, select the topic whose messages are to be queried. The messages stored in the topic are automatically displayed. You can also click **Advanced Search**, set search criteria, and click **Search** to search for messages.

**Search By:** method used for message query.

- **Creation time:** Messages are queried by creation time.
- **Offset:** Messages are queried by the location where the message is recorded in a partition.

 **NOTE**

If a broker is faulty and a topic with only one replica contains a large amount of data, an internal service error will be reported when you query messages in the topic. Shorten the time range for query based on the data volume.

**Table 8-3** Message content

| Parameter           | Description                                                                                                                                      |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Topic Name          | Name of the topic where the message is located.                                                                                                  |
| Message ID          | Message ID, which is determined by the value of <b>message_id</b> in the message header when a message is produced.                              |
| Application Key     | Application key of each message, which is determined by the value of <b>app_id</b> in the message header when a message is produced.             |
| Partition           | Partition where a message is located. The value starts from <b>0</b> . This parameter is mandatory if <b>Search By</b> is set to <b>Offset</b> . |
| Offset              | Offset of a message in a partition.                                                                                                              |
| Business Key        | Business key carried in a message, which identifies the service to which the message belongs.                                                    |
| Tag                 | Tags carried when a client produces messages to a topic. Generally, tags are not used.                                                           |
| Message Size (Byte) | Size of messages.                                                                                                                                |
| Created             | Time when a message was generated.                                                                                                               |

4. Click **Message Body** in a message record. In the dialog box that is displayed, view the message details.

## 8.5.2 Importing and Exporting Topics

### Overview

You can export consumer groups as a file to the local directory or batch import topics to ROMA Connect using a local file.

### Prerequisites

- Before importing a topic, ensure that the integration application to which the topic belongs has been created. Otherwise, [create an integration application](#) first.
- Before importing a topic, check whether a duplicate topic name exists in the instance to which the topic is to be imported. If there are duplicate topic names, the topic will fail to be imported.

- Ensure that the quota of topics meets the requirements before importing the topics.
- If the description of a topic contains a newline character, the newline character is escaped to \n in the exported CSV file of the topic. Therefore, if you import topics using such exported CSV files, manually modify the escape character \n back to a newline character.
- The file to be imported can contain a maximum of 100 topics. Otherwise, the topics cannot be imported.
- Do not edit the exported CSV file in Excel. Otherwise, the file cannot be used to import topics. To edit an exported file and then import it, export the file in XLSX or XLS format for editing.
- Topics that are not bound to any integration application cannot be exported.

## Importing Topics

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Message Queue Service > Topic Management**. On the page displayed, click **Import**.
3. In the dialog box displayed, select a local topic file and import it.
4. After the import is successful, you can view the imported topics in the topic list.

## Exporting Topics

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **Message Queue Service > Topic Management**.
3. Export topics.
  - Specified topics: Select the topics to be exported and choose **More > Export** to export a topic file to the local directory.
  - Exporting all topics: Click **Export All** to export a topic file to the local directory.
4. In the displayed dialog box, select the format of the file to be exported and export the topic file to the local directory.
  - Topics can be exported to XLSX, XLS, and CSV files. If the topic data contains commas (,), do not export the data to a CSV file because the CSV file will not be parsed properly.
  - In the exported file, the **Type** field indicates the permission type of the topic's integration application. **1** indicates that the integration application is the one to which the topic belongs. **2** indicates that the integration application has the publish permission for the topic while **3** indicates the subscribe permission.

For example, Topic1 belongs to App1 and has been granted permission for App2. Then the records of Topic1 are as follows:



- If App1 is configured with the publish and subscribe permissions, there are three Topic1 records about App1 in the exported file, and the values of **Type** are **1**, **2**, and **3**, respectively.
- If App1 is configured only with the publish or subscribe permission, there are two Topic1 records about App1 in the exported file, and the values of **Type** are **1** and **2**, or **1** and **3**.
- If App2 is configured with the publish and subscribe permissions, there are two Topic1 records about App2 in the exported file, and the values of **Type** are **2** and **3**, respectively.
- If App2 is configured only with the publish or subscribe permission, there is one Topic1 record about App2 in the exported file, and the value of **Type** is **2** or **3**.

## 8.6 Migrating Kafka Services

### Overview

Kafka service migration refers to the process of migrating the production and consumption message clients connected to other Kafka services to ROMA Connect and migrating some persistent message files to ROMA Connect.

During service migration, services that have high requirements on continuity must be smoothly migrated to the cloud because they cannot afford a long downtime.

### Preparations

1. Ensure that the network connections between the message production and consumption clients and the MQS connection address of the ROMA Connect instance are normal. You can view the MQS connection address on the **Instance Information** page of the ROMA Connect console.
  - If a private IP address is used for the connection, the clients and the ROMA Connect instance must be in the same VPC. If the clients and the ROMA Connect instance are in different VPCs, you can create a VPC peering connection to enable communication between the two VPCs. For details, see [VPC Peering Connection](#).
  - If a public network address is used for connection, the clients must have the permission to access the public network.
2. Ensure that the MQS specifications of the ROMA Connect instance cannot be lower than the Kafka specifications used by the original service. For details about MQS specifications, see [Product Specifications](#).
3. Create a topic with the same configurations as the original Kafka instance, including the topic name, number of replicas, number of partitions, message aging time, and whether to enable synchronous replication and flushing.

### Migration Scheme 1: Migrating the Production First

- **Solution**

In this solution, first migrate the message production service to ROMA Connect so that the original Kafka does not generate new messages. After all

messages in the original Kafka are consumed, migrate the message consumption service to ROMA Connect to consume new messages.

This is a common migration solution in the industry because the operation procedure is simple. The migration process is controlled by the service side. During the entire process, messages are not out of order. However, latency may occur because there is a period when you have to wait for all data to be consumed.

This scheme is applicable to **services that require the message sequence but are insensitive to the end-to-end latency.**

- **Migration Process**
  - a. Change the Kafka connection address of the production client to the MQS connection address of the ROMA Connect instance.
  - b. Restart the production service so that the producer can send new messages to the new ROMA Connect instance.
  - c. Check the consumption progress of each consumer group in the original Kafka instance until all data in the original Kafka instance is consumed.
  - d. Change the Kafka connection address of the consumer client to the MQS connection address of the ROMA Connect instance.
  - e. Restart the consumption service so that consumers can consume messages from the ROMA Connect instance.
  - f. Check whether consumers consume messages properly from the ROMA Connect instance.
  - g. The migration is completed.

## Migration Scheme 2: Migrating the Production Later

- **Solution**

Use multiple consumers for the consumption service. Some consume messages from the original Kafka instance, and others consume messages from the ROMA Connect instance. Then, migrate the production service to the ROMA Connect instance so that all messages can be consumed in time.

In this scheme, the consumption service may consume messages from the original Kafka and ROMA Connect at the same time in a period of time. Before the production service is migrated, the consumer service has been running on ROMA Connect. Therefore, there is no end-to-end latency problem. However, early on in the migration, data is consumed from both the original Kafka instance and ROMA Connect instance, so the messages may not be consumed in the order that they are produced.

This scheme is **suitable for services that require low latency but do not require strict message sequence.**
- **Migration Process**
  - a. Start new consumer clients, set the Kafka connection addresses to that of the ROMA Connect instance, and consume data from the ROMA Connect instance.

### NOTE

Original consumer clients must continue running. Messages are consumed from both the original Kafka instance and ROMA Connect instance.

- b. Modify the production client and change the Kafka connection address to the MQS connection address of the ROMA Connect instance.
- c. Restart the producer client to migrate the production service to the ROMA Connect instance.
- d. After the production service is migrated, check whether the consumption service connected to the ROMA Connect instance is normal.
- e. After all data in the original Kafka is consumed, close the original consumption clients.
- f. The migration is completed.

## Migrating Persistent Data

You can migrate consumed data from the original Kafka instance to the ROMA Connect instance by using the open-source tool [MirrorMaker](#). This tool mirrors the original Kafka consumer and ROMA Connect producer and migrates data to the ROMA Connect instance.

If the topic of the original Kafka contains a single replica and the topic of the ROMA Connect instance contains three replicas, it is recommended that the storage space of the ROMA Connect instance be three times that of the original Kafka.

# 9 Device Integration Guide

---

- [Usage Introduction](#)
- [Creating a Product](#)
- [Registering a Device](#)
- [Connecting Devices to ROMA Connect](#)
- [Product Management](#)
- [Device Management](#)
- [Rule Engine](#)
- [Subscription Management](#)

## 9.1 Usage Introduction

### Function Description

LINK is a component of ROMA Connect. It connects to devices for easy management on cloud through MQTT. ROMA Connect has the following advantages for device integration:

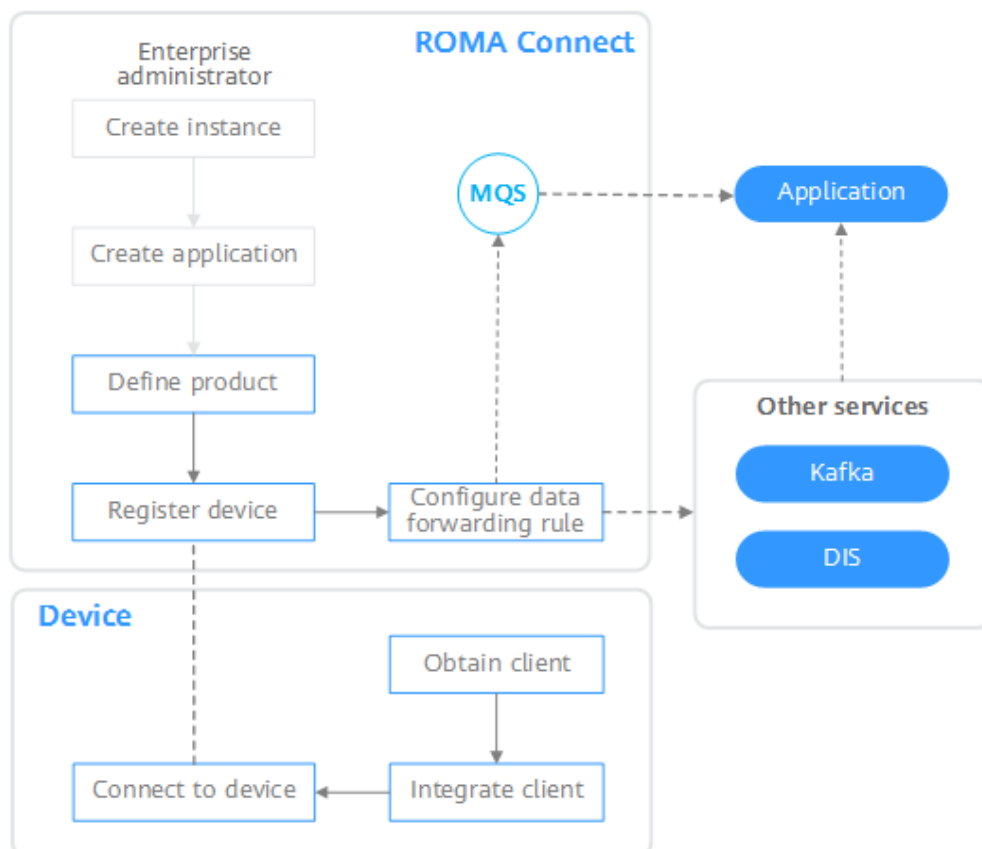
- **Standard MQTT protocols**  
The open-source standard MQTT SDK is used to connect devices to the cloud for message publishing and subscription.
- **Low-latency access for massive numbers of devices**  
LINK supports horizontal expansion of Broker and long connections of millions of devices.
- **Two-way synchronization between devices and applications**  
LINK implements two-way synchronization of configuration and status data between devices and applications.
- **Secure device information transmission**  
LINK provides authorization certification for devices and applications and bidirectional binding authorization for topics to ensure device security and

uniqueness. It provides TLS-based data transmission channels for secure message transmission.

## Process Flow

The following figure shows the process of using ROMA Connect for device integration.

**Figure 9-1** Process of using ROMA Connect for device integration



1. You have **created an instance and an integration application**.
2. **Create a product.**  
A product is a device model. You can define a product to determine the function attributes of devices. Each device belongs to a product.
3. **Register a device.**  
After a device is registered with ROMA Connect, a unique ID and key are allocated to the device to access LINK.
4. **Configure a data forwarding rule.**  
ROMA Connect does not directly store data reported by devices. You must configure data forwarding rules to forward device data to other services for storage. Data can be forwarded to ROMA MQS or other services such as Kafka and DIS.
5. **Connect to devices.**  
After being integrated with an MQTT client, a device can connect to ROMA Connect for reporting data and delivering commands.

## 9.2 Creating a Product

### Overview

A product is a device model, which is a set of service attributes of devices of the same type. Each device belongs to a product and inherits all service attributes of the product.

Product information includes basic information and thing model. It describes what a product is, what functions the product can provide, and what services the product can provide for external systems.

- Basic information

Describes basic information about a device, including the manufacturer name, manufacturer ID, product type, product model, protocol type, and device type. The product model and manufacturer ID are used together to identify a product.

For example, the product model of a water meter is **NBLoTDevice**, the manufacturer name is **HZYB**, the manufacturer ID is **TestUtf8Manuld**, and the device type is **WaterMeter**.

- Thing model

Service capabilities of a device need to be defined. After the capabilities of a device are divided into multiple thing model services, define the attributes, commands, and command parameters of each thing model service.

Take a water meter as an example. It has multiple capabilities, such as reporting data about the water flow, alarms, power, and connections, and receiving various commands delivered by a server. When describing the capabilities of a water meter, you can divide the capabilities into five thing model services, and define the attributes reported by each thing model service and their supported commands, as listed in the following table.

**Table 9-1** Service description

| Thing Model Service | Description                                                                                                                                                                                                                         |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| WaterMeterBasic     | Used to define parameters reported by the water meter, such as the water flow, temperature, and pressure. If these parameters need to be controlled or modified using commands, you also need to define parameters in the commands. |
| WaterMeterAlarm     | Used to define data reported by the water meter in various alarm scenarios. Commands need to be defined if necessary.                                                                                                               |
| Battery             | Used to define data including the voltage and current intensity of the water meter.                                                                                                                                                 |
| DeliverySchedule    | Used to define transmission rules for the water meter. Commands need to be defined if necessary.                                                                                                                                    |

| Thing Model Service | Description                                               |
|---------------------|-----------------------------------------------------------|
| Connectivity        | Used to define connection parameters for the water meter. |

 **NOTE**

The number of thing model services defined depends on the requirements. In the preceding example, the alarm thing model service can be split into the water pressure alarm service and traffic alarm service, or the alarm service can be integrated into the basic water meter service.

## Prerequisites

- Each product must belong to an integration application. Before creating a product, ensure that an integration application is available. Otherwise, [create an integration application](#) first.
- If you need to create a product by using a product template, [create a product template](#) first.

## Creating a Product

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Product Management**. On the page displayed, click **Create Product** in the upper right corner.
3. In the **Create Product** dialog box, set product parameters and click **OK**.

**Table 9-2** Parameters for creating a product

| Parameter     | Description                                                                                                                                                                                                                                                                                                              |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Product Type  | Select a product type. <ul style="list-style-type: none"><li>• <b>Common</b>: a directly connected device or a gateway subdevice.</li><li>• <b>Gateway</b>: a gateway device, which can be associated with subdevices and manage them.</li></ul>                                                                         |
| Protocol Type | Select the protocol type used by the product. <ul style="list-style-type: none"><li>• <b>MQTT</b>: a publish-subscribe network protocol that transports messages between devices.</li><li>• <b>Modbus</b>: a serial communication protocol.</li><li>• <b>OPC UA</b>: an Ethernet-based communication protocol.</li></ul> |
| Product Name  | Enter a product name. It is recommended that you enter a name based on naming rules to facilitate search.                                                                                                                                                                                                                |

| Parameter               | Description                                                                                                                                                                                                                                                                                |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Product Template        | After a product template is selected, the created product has all services and attributes of the product template.<br>Only product templates in the <b>Enabled</b> state can be selected.                                                                                                  |
| Integration Application | Select the integration application to which the product belongs.                                                                                                                                                                                                                           |
| Manufacturer Name       | Enter the manufacturer name of the device in use.                                                                                                                                                                                                                                          |
| Manufacturer ID         | Enter the manufacturer ID of the device in use.                                                                                                                                                                                                                                            |
| Product Model           | Enter the product model of the device in use.<br>The product model and manufacturer ID collectively identify a product. The combination of product model and manufacturer ID must be unique.                                                                                               |
| Device Type             | Select the type of the device to be connected. You can select <b>Default Type</b> or add a device type that is added in <b>DEVICE_TYPE</b> on the <b>Data Dictionaries</b> tab page of the <b>Instance Information</b> page. For details, see <a href="#">Creating a Data Dictionary</a> . |
| Model Version           | Enter a custom model version of the device.                                                                                                                                                                                                                                                |
| Description             | Enter a brief description of the product.                                                                                                                                                                                                                                                  |

## (Optional) Adding a Thing Model Service for a Product

1. On the ROMA Connect console, choose **LINK > Product Management** and click a product name to access the product details page.
2. Add thing model services for the product.
  - a. On the **Thing Model** tab page, click **Create**.
  - b. In the **Create Thing Model Service** dialog box, set thing model service parameters and click **OK**.

**Table 9-3** Parameters for creating a thing model service

| Parameter | Description                                                                                                           |
|-----------|-----------------------------------------------------------------------------------------------------------------------|
| Name      | Enter a thing model service name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Status    | This parameter specifies whether to enable the thing model service. The thing model service is enabled by default.    |



| Parameter   | Description                                           |
|-------------|-------------------------------------------------------|
| Description | Enter a brief description of the thing model service. |

3. Add attributes for the thing model service.
  - a. On the **Thing Model** tab page, select the thing model service for which you want to add attributes.
  - b. On the **Attributes** tab page, click **Create**.
  - c. In the **Create Attribute** dialog box, set attribute parameters and click **OK**.

**Table 9-4** Parameters for creating an attribute


| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Enter an attribute name. It is recommended that you enter a name based on naming rules to facilitate search.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Data Type   | <p>Select the data type of the attribute.</p> <ul style="list-style-type: none"> <li>• <b>Int</b>: integer. If you select <b>Int</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, <b>Step</b>, and <b>Unit</b>.</li> <li>• <b>Number</b>: number. If you select <b>Number</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, <b>Step</b>, and <b>Unit</b>.</li> <li>• <b>String</b>: a character string. If you select <b>String</b>, you also need to specify the maximum data length and enumerated values.</li> <li>• <b>Bool</b>: Boolean. If you select <b>Bool</b>, you also need to specify <b>0</b> and <b>1</b>.</li> <li>• <b>DateTime</b>: date. If you select <b>DateTime</b>, you also need to specify the maximum data length.</li> <li>• <b>JsonObject</b>: JSON object. If you select <b>JsonObject</b>, you also need to specify the maximum data length.</li> <li>• <b>Array</b>: array.</li> </ul> <p>The following uses the <b>Int</b> type as an example:<br/>When defining the temperature attribute for thermometers, set <b>Type</b> to <b>Int</b>, <b>Min. Value</b> to <b>0</b>, <b>Max. Value</b> to <b>100</b>, <b>Step</b> to <b>2</b>, and <b>Unit</b> to <b>°C</b>. This indicates that the device reports the temperature value (for example, 0°C, 2°C, 4°C, 6°C and 8°C) each time the temperature changes by two degrees.</p> |
| Mandatory   | This parameter specifies whether the attribute must be reported by a device. This parameter is enabled by default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Description | Enter a brief description of the attribute.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |

| Parameter                                                                                                                               | Description                                                                                                                           |
|-----------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| Min. Value<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example. | Minimum value of the thing model service attribute.                                                                                   |
| Max. Value<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example. | Maximum value of the thing model service attribute.                                                                                   |
| Step<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example.       | Step of the attribute.                                                                                                                |
| Unit<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example.       | Enter the unit of the attribute. The value cannot exceed 50 characters. For example, the unit of the temperature attribute can be °C. |

4. Add commands for the thing model service.
  - a. On the **Thing Model** tab page, select the thing model service for which you want to add commands.
  - b. On the **Commands** tab page, click **Create**.
  - c. In the **Create Command** dialog box, set command parameters and click **OK**.

**Table 9-5** Parameters for creating a command

| Parameter    | Description                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------------------|
| Command Name | Enter a command name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Description  | Enter a brief description of the command.                                                                 |

- d. Locate the created command in the command list and click  next to the command name to expand the command field list.  
Click **Delivery Command Fields** or **Response Command Fields** on the right to view the delivery command fields and response command fields of the command.
- e. Click **Create Field**.
- f. In the **Create Delivery Command Field** or **Create Response Command Field** dialog box, set field parameters and click **OK**.

**Table 9-6** Parameters for creating a field

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field Name  | Enter a field name. It is recommended that you enter a name based on naming rules to facilitate search.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Data Type   | Select the data type of the command field. <ul style="list-style-type: none"><li>● <b>Int</b>: integer. If you select <b>Int</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, <b>Step</b>, and <b>Unit</b>.</li><li>● <b>Number</b>: number. If you select <b>Number</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, <b>Step</b>, and <b>Unit</b>.</li><li>● <b>String</b>: a character string. If you select <b>String</b>, you also need to specify the maximum data length and enumerated values.</li><li>● <b>Bool</b>: Boolean. If you select <b>Bool</b>, you also need to specify <b>0</b> and <b>1</b>.</li><li>● <b>DateTime</b>: date. If you select <b>DateTime</b>, you also need to specify the maximum data length.</li><li>● <b>JsonObject</b>: JSON object. If you select <b>JsonObject</b>, you also need to specify the maximum data length.</li><li>● <b>Array</b>: array.</li></ul> |
| Mandatory   | This parameter specifies whether the field must be carried in the delivery command. By default, the field must be carried.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Description | Enter a brief description of the command field.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

| Parameter                                                                                                                               | Description                                                                                                                               |
|-----------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| Min. Value<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example. | Minimum value of the delivery command field.                                                                                              |
| Max. Value<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example. | Maximum value of the delivery command field.                                                                                              |
| Step<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example.       | Step of the delivery command field.                                                                                                       |
| Unit<br><b>NOTE</b><br>The configuration items are displayed based on the selected data type. The Int type is used as an example.       | Enter the unit of the delivery command field. The value cannot exceed 50 characters. For example, the unit of the voltage field can be V. |

### (Optional) Adding a Customized Topic for a Product

If the basic topics of a device cannot meet your requirements, you can customize a topic for the device. After the customized topic is added to a product, the topic is inherited by all devices under the product.

1. On the ROMA Connect console, choose **LINK > Product Management** and click a product name to access the product details page.
2. On the **Topic Classes** tab page, click **Create Topic**.
3. In the dialog box displayed, configure topic information and click **OK**.

**Table 9-7** Parameters for creating a topic

| Parameter            | Description                                                                                                                                                                                                                                                                                                           |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Operation Permission | Select the topic permission granted for the device. <ul style="list-style-type: none"><li>• <b>Publishing:</b> The device has the permission to publish messages into the topic.</li><li>• <b>Subscription:</b> The device has the permission to subscribe to the topic and obtain messages from the topic.</li></ul> |
| Topic Class Name     | Enter the custom field in the name of the custom topic. The version number, device ID, and customized fields form a custom topic name: /{Version number}/{Device ID}/{Custom field}.                                                                                                                                  |
| Version Number       | Enter the version number of the topic, for example, <b>V1.0</b> .                                                                                                                                                                                                                                                     |
| Description          | Enter the brief description of the custom topic.                                                                                                                                                                                                                                                                      |

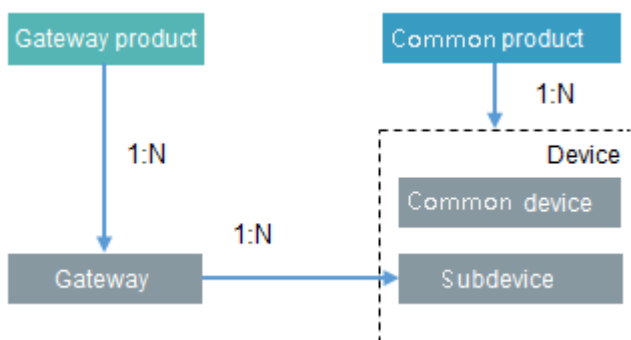
## 9.3 Registering a Device

### Overview

After a device is registered with ROMA Connect, a unique ID and password are allocated to the device in the system. The device uses the ID and password to access ROMA Connect.

A device that is directly connected to ROMA Connect is called common device, and a device that is connected to ROMA Connect through a gateway is called subdevice.

The following figure shows the relationship between devices and products. Both common devices and subdevices belong to common products. Gateways belong to gateway products. A gateway can connect to multiple subdevices.



### Prerequisites

- Each device must belong to an integration application. Before registering a device, ensure that an integration application is available. Otherwise, **create an integration application** first.

- Each device must belong to a product. Before registering a device, ensure that a product is available. Otherwise, [create a product](#) first.

## Creating a Device

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **LINK > Device Management**. On the page displayed, click **Create Device** in the upper right corner.
3. In the **Create Device** dialog box, set device parameters and click **OK**.

**Table 9-8** Parameters for creating a device

| Parameter               | Description                                                                                                                                                                                                                                          |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Device Name             | Enter a device name. It is recommended that you enter a name based on naming rules to facilitate search.                                                                                                                                             |
| Integration Application | Select the integration application to which a device belongs.                                                                                                                                                                                        |
| Associated Product      | Select the product to which the device belongs. The device inherits all thing model service capabilities defined for the product.<br>For common devices and subdevices, select a common product. For gateways, select a gateway product.             |
| Device ID               | Enter the unique identifier of the device, such as the IMEI and MAC address. The device ID corresponds to the deviceId of a directly connected device or a subdevice under a gateway or corresponds to the gatewayId of a gateway.                   |
| Password                | Enter the password for connecting the device to ROMA Connect. The password is user-defined.                                                                                                                                                          |
| Confirm Password        | Enter the confirm password, which must be the same as the value of <b>Password</b> .                                                                                                                                                                 |
| Status                  | This parameter specifies whether to enable the device. The device can connect to ROMA Connect only after being enabled. By default, the device is enabled.                                                                                           |
| Device Tag              | Add tags for the device to facilitate search.<br><b>NOTE</b><br>When editing a tag, if a message indicating that the tag already exists is mistakenly displayed, delete all tags on the editing page, add the tags again, and save the modification. |
| Description             | Enter a brief description of the device.                                                                                                                                                                                                             |

## (Optional) Adding a Custom Topic for a Device

If the basic topics of a device cannot meet your requirements, you can customize a topic for the device.

1. On the ROMA Connect console, choose **LINK > Device Management** and click a device name to access the device details page.
2. On the **Topics** tab page, select **Custom Topics** and click **Create** on the right.
3. In the dialog box displayed, set custom topic parameters and click **OK**.

**Table 9-9** Parameters for creating a custom topic

| Parameter   | Description                                                                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Enter a custom topic name.                                                                                                                                                                                                                                       |
| Permission  | Select the topic permission granted for the device. <ul style="list-style-type: none"><li>● <b>Publish</b>: The device can publish messages to the topic.</li><li>● <b>Subscribe</b>: The device can subscribe to the topic and read messages from it.</li></ul> |
| Description | Enter a brief description of the custom topic.                                                                                                                                                                                                                   |

## 9.4 Connecting Devices to ROMA Connect

### 9.4.1 Connecting MQTT Devices

#### Overview

A device can access ROMA Connect by integrating with the MQTT.fx client. After a device is connected to ROMA Connect, it can report data to ROMA Connect. You can also use ROMA Connect to deliver control commands to the device.

ROMA Connect does not directly store data reported by devices. You must [configure data forwarding rules](#) to forward device data to other services for storage.

---

#### NOTICE

When MQTT devices are used, only QoS0 and QoS1 in MQTT are supported.

---

#### Prerequisites

Devices can communicate with a ROMA Connect instance over the public network. Ensure that the instance has been bound with an EIP.

## Connecting a Device

A device can access ROMA Connect using the open-source Eclipse Paho MQTT client. You can also use the [MQTT X client](#) to debug device access.

1. Obtain the MQTT client.  
Obtain the [Eclipse Paho MQTT client](#) based on your programming language.
2. Obtain device access information.
  - a. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
  - b. In the navigation pane on the left, choose **LINK > Device Management**.
  - c. Obtain device connection information.
    - MQTT/MQTTS connection address: Obtain the access address of the device from the upper part of the **Device Management** page and then download the SSL certificate. If MQTT is used for access, obtain the MQTT connection address. If MQTTS is used for access, obtain the MQTTS connection address.
    - Client ID/Username/Password: Locate the corresponding device on the **Devices** tab page and obtain the client ID, username, and password of the device.
  - d. Obtain topic information of the device.

Click the device name to go to the device details page. On the **Topics** tab page, obtain the topic information about the messages reported (published) and received (subscribed) by the device.

After the device is created, the system generates five basic topics. For details about the topics, see [Table 9-10](#). For details about how to use the topics, see [MQTT Topic Specifications](#).

**Table 9-10** Basic topic description

| Topic Name                               | Topic Class | Description                                                                                              |
|------------------------------------------|-------------|----------------------------------------------------------------------------------------------------------|
| <i>{Product ID}/out/<br/>{Device ID}</i> | Publish     | Used by devices to publish messages. Messages are processed based on the rule engine configuration.      |
| <i>{Product ID}/in/<br/>{Device ID}</i>  | Subscribe   | Used by devices to subscribe to messages. Messages are processed based on the rule engine configuration. |



| Topic Name                                           | Topic Class | Description                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/v1/devices/{Device ID}/datas</code>           | Publish     | Used by devices to report messages complying with the MQTT-based communication protocol for IoT in power distribution. Messages are filtered based on the protocol and then processed based on the rule engine configuration. When the device shadow function is enabled, the data reported by this topic will be recorded in the shadow. |
| <code>/v1/devices/{Device ID}/command</code>         | Subscribe   | Used by devices to subscribe to commands delivered by the IoT platform. Commands delivered by the IoT platform are published to this topic by default.                                                                                                                                                                                    |
| <code>/v1/devices/{Device ID}/commandResponse</code> | Publish     | Used by devices to publish responses. After a device receives a command from the IoT platform, it responds to the command through this topic.                                                                                                                                                                                             |

3. Integrate the device with the MQTT client.  
Develop and integrate the device with Eclipse Paho MQTT client, and write the device access information during the integration. For details, see [Configuring Device Integration](#).
4. Connect the device to ROMA Connect.  
After the integration development is complete, power on the device to take it online and connect it to ROMA Connect.

## Debugging a Device

- Reporting data to ROMA Connect  
By default, ROMA Connect has subscribed to topics of the publish type. After a device sends a message to such a topic, the topic can report data to ROMA Connect.
- Receiving commands from ROMA Connect  
After ROMA Connect sends a control command to the command delivery topic, the topic delivers the command to the device for device control.

## 9.4.2 Connecting Modbus Devices

### Overview

A device can connect to ROMA Connect by integrating EdgeShell. EdgeShell is used to collect data from edge devices and send the data to the edge gateway of

ROMA Connect. EdgeShell supports the access of devices that use the Modbus or OPC UA protocol. This section describes how to connect devices using the Modbus protocol.

ROMA Connect does not directly store data reported by devices. You must [configure data forwarding rules](#) to forward device data to other services for storage.

## Prerequisites

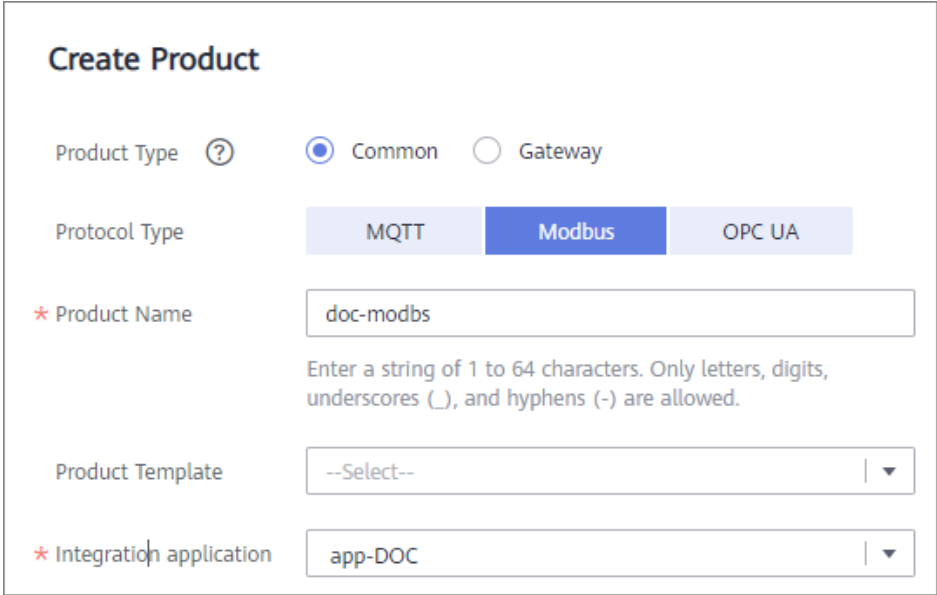
Devices can communicate with a ROMA Connect instance over the public network. Ensure that the instance has been bound with an EIP.

## Connecting a Device

EdgeShell is used to collect data from edge devices and send the data to the edge gateway of ROMA Connect. To connect a device using Modbus, perform the following steps:

1. Create a product.

On the **Product Management** page, click **Create Product**, set **Product Type** to **Common**, and set **Protocol Type** to **ModBus**.



**Create Product**

Product Type ?  Common  Gateway

Protocol Type MQTT **Modbus** OPC UA

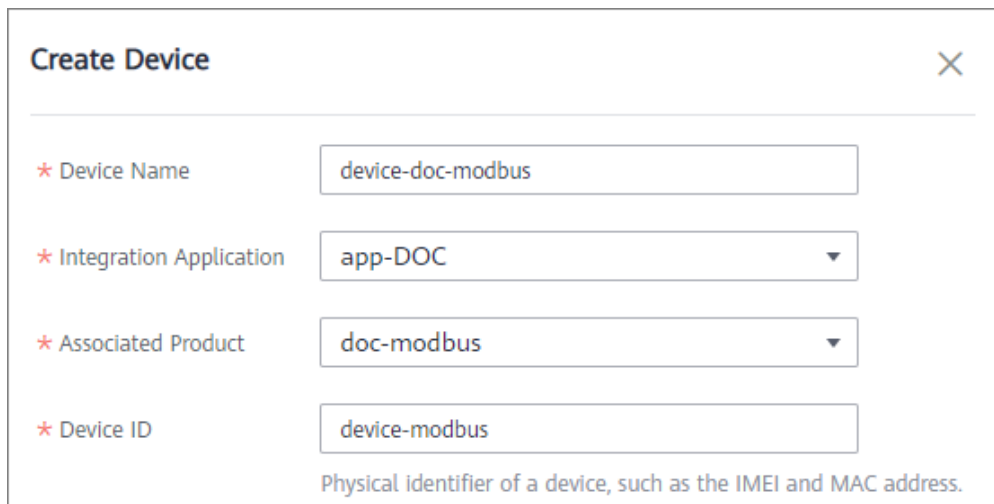
\* Product Name   
Enter a string of 1 to 64 characters. Only letters, digits, underscores (\_), and hyphens (-) are allowed.

Product Template

\* Integration application

2. Create a device.

On the **Device Management** page, click **Create Device** and select the product created in [1](#).



**Create Device** ✕

\* Device Name

\* Integration Application

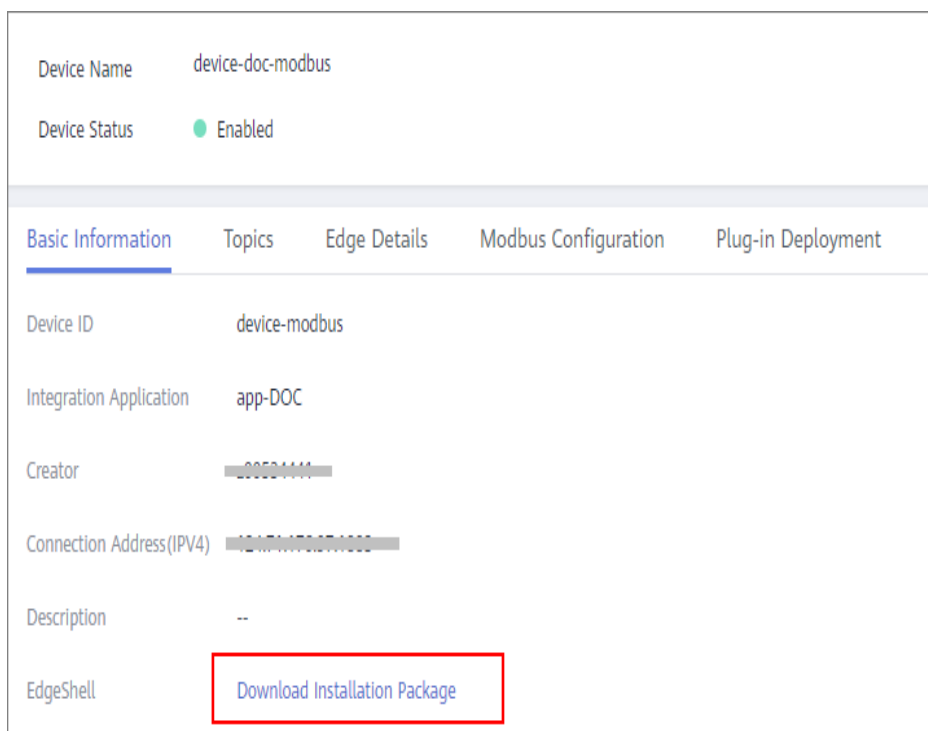
\* Associated Product

\* Device ID

Physical identifier of a device, such as the IMEI and MAC address.

3. Download the EdgeShell installation package.

Click the device name to access the device details page. On the **Basic Information** tab page, click **Download Installation Package** next to EdgeShell to download the EdgeShell installation package.



Device Name device-doc-modbus

Device Status ● Enabled

**Basic Information** Topics Edge Details Modbus Configuration Plug-in Deployment

Device ID device-modbus

Integration Application app-DOC

Creator XXXXXXXXXX

Connection Address(IPV4) XXXXXXXXXX

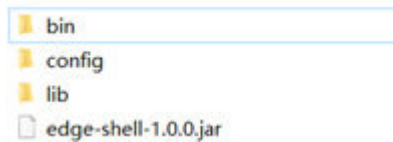
Description --

EdgeShell Download Installation Package

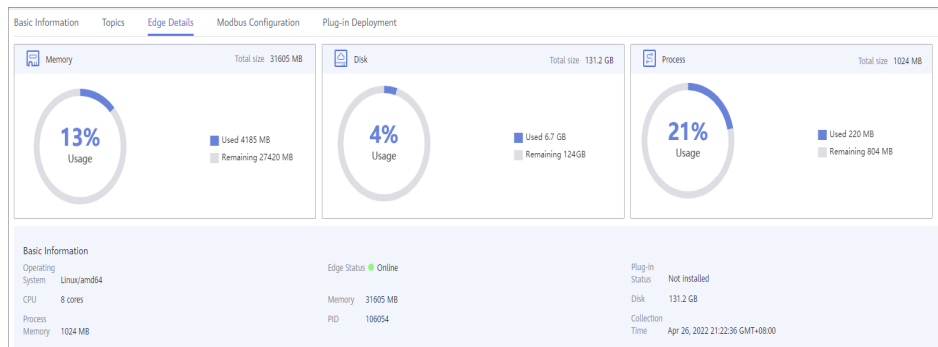
4. Start EdgeShell.

EdgeShell can run in both the Windows and Linux environments. The startup script varies according to the environment.

- a. EdgeShell depends on the Java 11 operating environment. Before running EdgeShell, ensure that the Java operating environment has been installed and the **JAVA\_HOME** environment variable has been configured.
- b. Decompress the downloaded **edge-shell.zip** package. The following figure shows the directory structure obtained after the decompression.



- **bin**: startup and stop scripts
  - **config**: configuration parameters
  - **lib**: dependent jar package
  - **edge-shell-1.0.0.jar**: main program
- c. On the **Basic Information** tab page of the device details page, copy **SSL Connection Address(IPv4)** to the **mqtt.properties** file in the **config** directory.
  - d. Run the startup script.
    - In Windows, run **bin/windows/start.bat**.
    - In Linux, run **bin/linux/start.sh**.
  - e. After the device is started, the EdgeShell is connected to LINK. In this case, the device is displayed as online, and data is displayed in the **Edge Details** tab page of the device details page.



**NOTE**

After the script is started, the **log** directory is generated, which records startup and run logs. The **edge-shell-error.log** and **edge-shell-info.log** files exist in the directory. If the device is not connected after the startup, you can view the error logs to quickly locate the fault.

On Windows, if the log directory is not generated after you click **Start**, modify the following statement in the **start.bat** file.

Before the change:

```
start edge-shell %JAVA_OPT% -jar edge-shell-1.0.0.jar
```

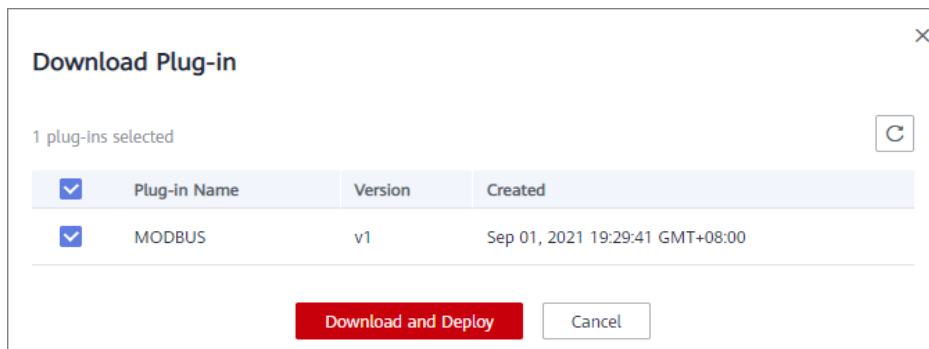
pause

After the change:

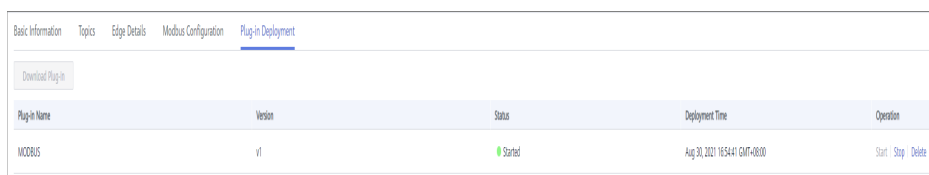
```
java %JAVA_OPT% -jar edge-shell-1.0.0.jar
```

5. Download the plug-in.

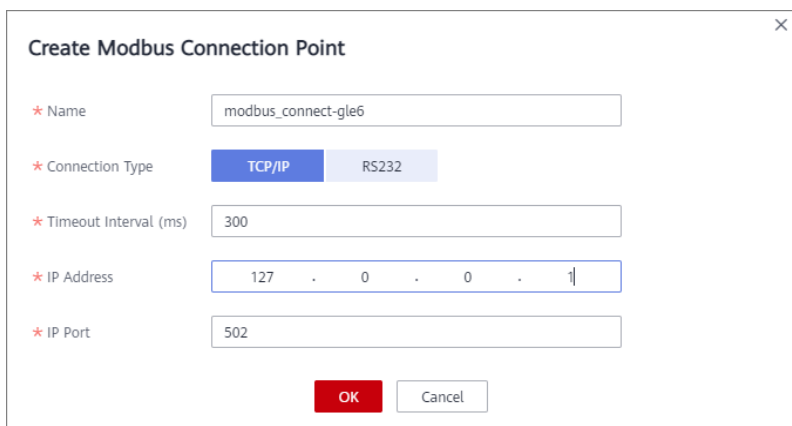
On the **Plug-in Deployment** page, click **Download Plug-in**. In the displayed dialog box, click **Download and Deploy**.



After the plug-in is installed, click **Start** in the **Operation** column to start the plug-in.



6. Configure the Modbus connection point and collection policy.
  - a. If the created device is a Modbus device, **Modbus Configuration** is displayed on the device details page. The Modbus plug-in supports TCP/IP and serial port connection modes. The configuration is as follows:
    - To create a Modbus connection point with the TCP/IP type, set the parameters, as shown in the following figure.



- To create a Modbus connection point with the serial type, set the parameters, as shown in the following figure.

b. Create a collection policy under the created connection point.

| Register Point | Attribute Name | Operation |
|----------------|----------------|-----------|
| ⊕ Add Point    |                |           |

**NOTE**

The register address of a user device is determined by the start address and function code.

The Modbus protocol defines the range of a device address (or start address) to be 0 to 65535, while the register address ranges from 1 to 65536. Therefore, if the start address is 5, its register address is 6.

Function codes 01, 02, 03, and 04 are supported. Range of their register address:

01: 000001 to 065536

02: 100001 to 165536

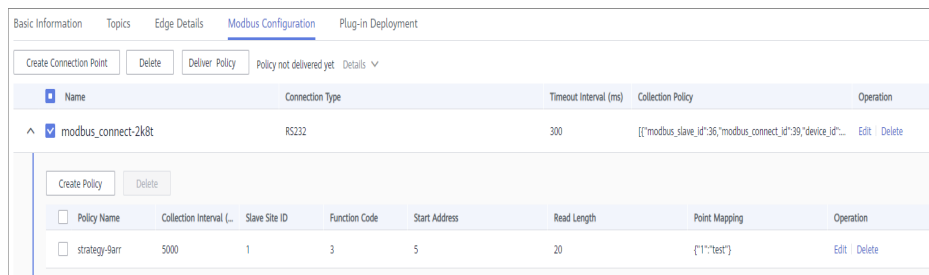
03: 400001 to 465536

04: 300001 to 365536

For example, if the function code is 03 and the start address is 5, the corresponding register address is 400006.

7. Deliver the collection policy to the EdgeShell.

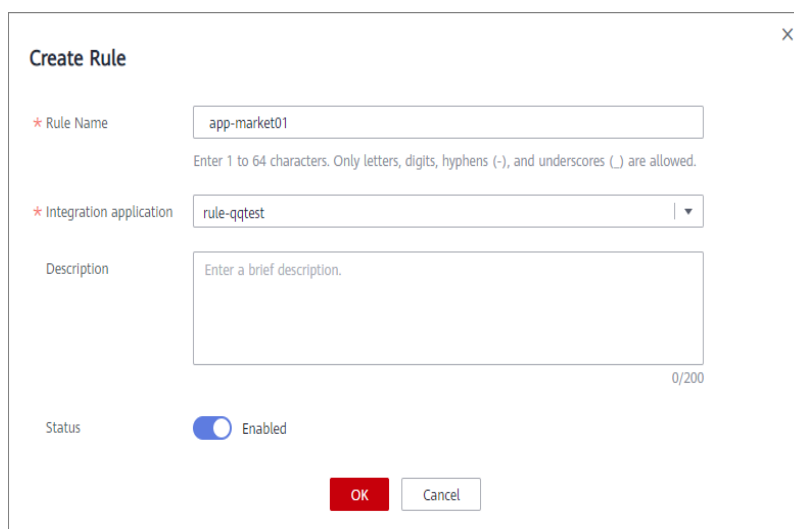
On the **Modbus Configuration** tab page of the device details page, select the target Modbus connection point and click **Deliver Policy**.



8. Configure the rule engine.

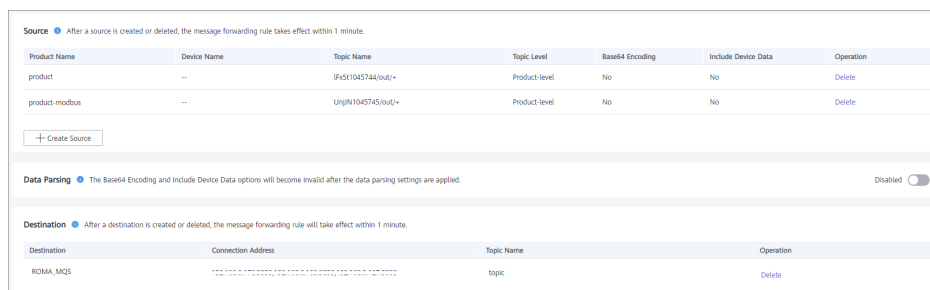
Forward data collected from common devices to MQS.

- a. On the **Rule Engine** page, click **Create Rule**. In the dialog box displayed, select the same application as that of the common product.



- b. Click the rule engine name to access the rule engine details page and configure the source and destination.

- Select the created product and device as the data source.
- Select MQS in the same instance as the destination.



## Debugging a Device

After the rule engine forwards the collected data to MQS, you can view the reported data on MQS. In this case, the data is reported to ROMA Connect.

## 9.4.3 Connecting OPC UA Devices

### Overview

A device can connect to ROMA Connect by integrating EdgeShell. EdgeShell is used to collect data from edge devices and send the data to the edge gateway of ROMA Connect. EdgeShell supports the access of devices that use the Modbus or OPC UA protocol. This section describes how to connect devices using the OPC UA protocol.

ROMA Connect does not directly store data reported by devices. You must [configure data forwarding rules](#) to forward device data to other services for storage.

### Prerequisites

Devices can communicate with a ROMA Connect instance over the public network. Ensure that the instance has been bound with an EIP.

### Connecting a Device

1. Create a product.

On the **Product Management** page, click **Create Product**, set **Product Type** to **Common**, and set **Protocol Type** to **OPC UA**.

**Create Product**

Product Type  Common  Gateway

Protocol Type MQTT Modbus OPC UA

\* Product Name   
Enter a string of 1 to 64 characters. Only letters, digits, underscores (\_), and hyphens (-) are allowed.

Product Template

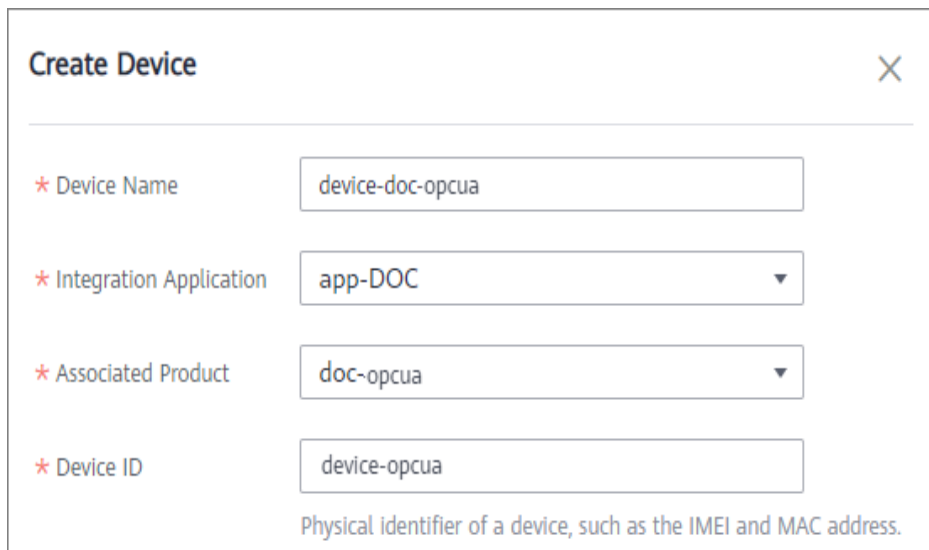
\* Integration Application

\* Manufacturer Name   
Enter a string of 2 to 64 characters.

2. Create a device.

On the **Device Management** page, click **Create Device** and select the product created in [1](#).





**Create Device** [X]

\* Device Name

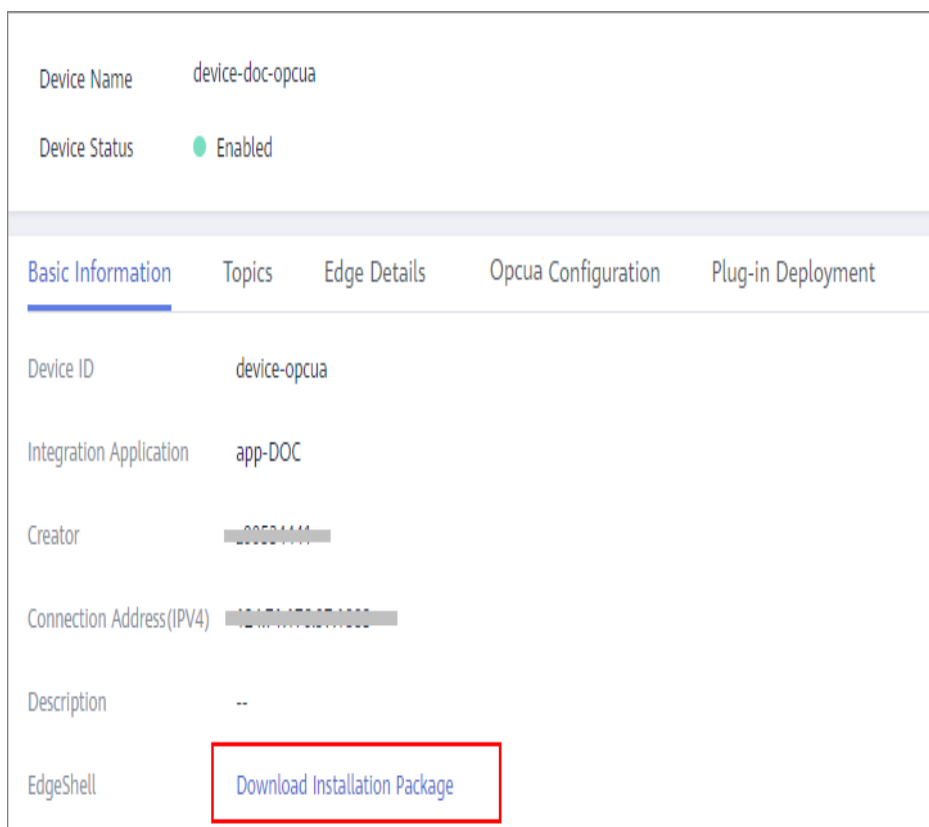
\* Integration Application

\* Associated Product

\* Device ID

Physical identifier of a device, such as the IMEI and MAC address.

3. Download the EdgeShell installation package.  
Click the device name to access the device details page. On the **Basic Information** tab page, click **Download Installation Package** next to EdgeShell to download the EdgeShell installation package.



Device Name device-doc-opcua

Device Status ● Enabled

**Basic Information** Topics Edge Details Opcua Configuration Plug-in Deployment

Device ID device-opcua

Integration Application app-DOC

Creator [REDACTED]

Connection Address(IPV4) [REDACTED]

Description --

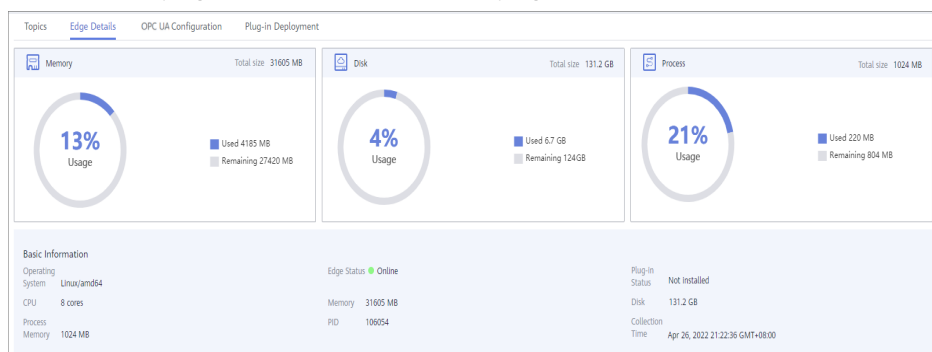
EdgeShell [Download Installation Package](#)

4. Start EdgeShell.  
EdgeShell can run in both the Windows and Linux environments. The startup script varies according to the environment.
  - a. EdgeShell depends on the Java 11 operating environment. Before running EdgeShell, ensure that the Java operating environment has been installed and the **JAVA\_HOME** environment variable has been configured.

- b. Decompress the downloaded **edge-shell.zip** package. The following figure shows the directory structure obtained after the decompression.



- bin: startup and stop scripts
  - config: configuration parameters
  - lib: dependent jar package
  - edge-shell-1.0.0.jar: main program
- c. On the **Basic Information** tab page of the device details page, copy **SSL Connection Address(IPv4)** to the **mqtt.properties** file in the **config** directory.
- d. Run the startup script.
- In Windows, run **bin/windows/start.bat**.
  - In Linux, run **bin/linux/start.sh**.
- e. After the device is started, the EdgeShell is connected to LINK. In this case, the device is displayed as online, and data is displayed in the **Edge Details** tab page of the device details page.



#### NOTE

After the script is started, the **log** directory is generated, which records startup and run logs. The **edge-shell-error.log** and **edge-shell-info.log** files exist in the directory. If the device is not connected after the startup, you can view the error logs to quickly locate the fault.

On Windows, if the log directory is not generated after you click **Start**, modify the following statement in the **start.bat** file.

Before the change:

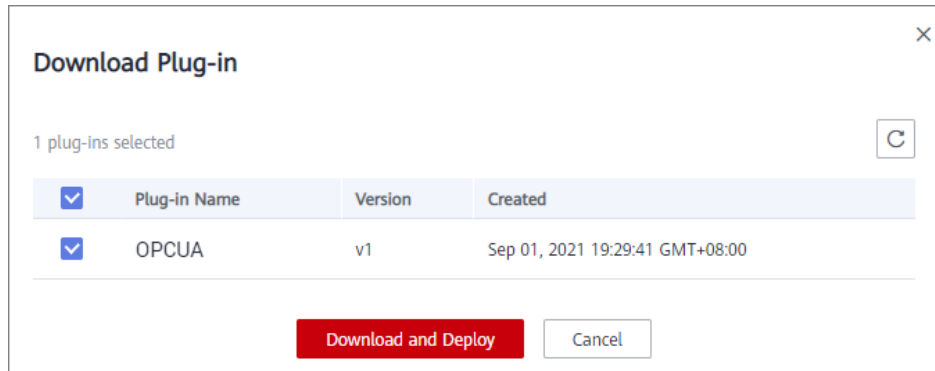
```
start edge-shell %JAVA_OPT% -jar edge-shell-1.0.0.jar
```

pause

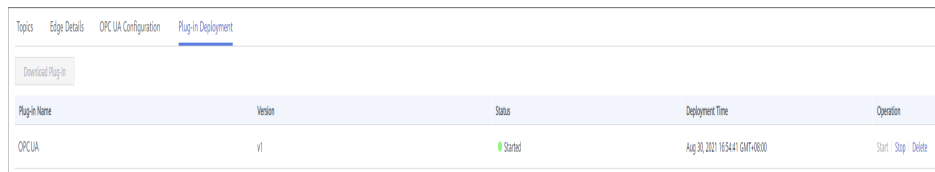
After the change:

```
java %JAVA_OPT% -jar edge-shell-1.0.0.jar
```

5. Download the plug-in.
- On the **Plug-in Deployment** page, click **Download Plug-in**. In the displayed dialog box, click **Download and Deploy**.



After the plug-in is installed, click **Start** in the **Operation** column to start the plug-in.



6. Configure the OPC UA connection point and collection policy.
  - a. Create an OPC UA connection point. Set **Collection** to **as scheduled** or **as subscribed**, **Security Policy** to **None**, **Basic128Rsa15**, **Basic256**, **Basic256Sha256**, **Aes128\_Sha256\_RsaOaep**, or **Aes256\_Sha256\_RsaPss**, and **Security Mode** to **None** or **Username/Password**.

**Figure 9-2** Setting Collection to as scheduled

**Create OPC UA Connection Point**

\* Name

\* TCP Connection Address

\* Collection

\* Collection Interval (ms)

\* Regular Collection

\* Security Policy

\* Security Mode

\* Username

\* Password

\* Timeout Interval (ms)

\* Topic Name

**Figure 9-3** Setting Collection to as subscribed

**Create OPC UA Connection Point**

- Name: opcua\_connect-tejl
- TCP Connection Address: opc.tcp://
- Collection: as subscribed
- Security Policy: Basic256Sha256
- Security Mode: User Authentication
- Username: test
- Password: .....
- Timeout Interval (ms): 200
- Topic Name: e91CY1csX73/out/opcua

Buttons: OK, Cancel

- b. Create a node under the created connection point and enter the node path.

**Create Node**

- Node Path: s=HelloWorld/ScalarTypes/int16
- Node Description: Enter a node description. (0/200)

Buttons: OK, Cancel

7. Deliver the collection policy.

On the **OPC UA Configuration** tab page of the device details page, select the target OPC UA connection point and click **Deliver Policy**.

| OPC UA Configuration |                              |                  |                 |          |                     |                    |                   |                 |             |
|----------------------|------------------------------|------------------|-----------------|----------|---------------------|--------------------|-------------------|-----------------|-------------|
| Name                 | Connection Point Address     | Timeout Interval | Security Policy | Username | Collection Interval | Regular Collection | Topic Name        | Collection Node | Operation   |
| opcua_connect-ak...  | opc.tcp://192.168.1.100:4840 | 200              | None            | ..       | 5000                | No                 | UnjN1045745(ou... |                 | Edit Delete |

| Node Path                      | Created                         | Operation   |
|--------------------------------|---------------------------------|-------------|
| s=HelloWorld/ScalarTypes/int16 | Sep 13, 2021 09:54:45 GMT+08:00 | Edit Delete |

8. Configure the rule engine.

Forward data collected from common devices to MQS.

- a. On the **Rule Engine** page, click **Create Rule**. In the dialog box displayed, select the same application as that of the common product.

- b. Click the rule engine name to access the rule engine details page and configure the source and destination.
  - Select the created product and device as the data source.
  - Select MQS in the same instance as the destination.

**Source** After a source is created or deleted, the message forwarding rule takes effect within 1 minute.

| Product Name  | Device Name | Topic Name        | Topic Level   | Base64 Encoding | Include Device Data | Operation |
|---------------|-------------|-------------------|---------------|-----------------|---------------------|-----------|
| product       | --          | IFdS1045744/out/+ | Product-level | No              | No                  | Delete    |
| product-mobus | --          | UngN1045745/out/+ | Product-level | No              | No                  | Delete    |

[+ Create Source](#)

**Data Parsing** The Base64 Encoding and Include Device Data options will become invalid after the data parsing settings are applied. Disabled

**Destination** After a destination is created or deleted, the message forwarding rule will take effect within 1 minute.

| Destination | Connection Address | Topic Name | Operation |
|-------------|--------------------|------------|-----------|
| ROMA_MQS    | .....              | topic      | Delete    |

## Debugging a Device

After the rule engine forwards the collected data to MQS, you can view the reported data on MQS. In this case, the data is reported to ROMA Connect.

# 9.5 Product Management

## 9.5.1 Viewing Product Information


### Overview

After a product is created, you can view and reset the password, and import and export the product.

## Functions

After a product is created, you can view and edit the product as required by referring to the following table.

**Table 9-11** Function description

| Function                            | Description                                                                                                                                                                                                                                                                                                                                                                      |
|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Viewing passwords                   | <p>A product password is a credential for accessing all the devices of a product.</p> <p>Click  in the <b>Password</b> column of a product to view its password.</p>                                                                                                                            |
| Importing products                  | <p>If a file already exists on the local host, you can click <b>Import</b> on the <b>Products</b> tab page to import the file that contains product information.</p> <p>To obtain a file template, create a product on the console, select the product, and click <b>Export</b> to export a CSV file. Then, you can modify the CSV file based on the product to be imported.</p> |
| Exporting specified or all products | <p>Export information about one or more products to a CSV file.</p> <p>To export one or more products, select the products to be exported and click <b>Export</b>. To export all products, click <b>Export All</b>.</p>                                                                                                                                                          |
| Resetting a password                | <p>To change a password, click <b>Reset Password</b> in the <b>Operation</b> column of a product.</p>                                                                                                                                                                                                                                                                            |
| Editing a product                   | <p>Click <b>Edit</b> in the <b>Operation</b> column of a product. <b>Product Type</b>, <b>Protocol Type</b>, and <b>Application</b> cannot be modified.</p>                                                                                                                                                                                                                      |
| Deleting one or more products       | <p>Delete a created product.</p> <p>To delete one or more products, select the required products and click <b>Delete</b>. You can also click <b>Delete</b> in the <b>Operation</b> column of a specific product to delete it.</p>                                                                                                                                                |

## 9.5.2 Importing and Exporting Products

### Overview

ROMA Connect supports product import and export. You can export products as a file to the local host or import a local product file to ROMA Connect to migrate product information in batches.

### Prerequisites

- Before importing a product, ensure that the integration application to which the product belongs has been created. Otherwise, [create the integration application](#) in advance.

- Before importing a product, check whether duplicate product names exist in the instance to which the product is to be imported. If duplicate product names exist, the product fails to be imported.
- Ensure that the quota of products meets the requirements before importing the products.
- The files to be imported must be in CSV format and have been encoded using UTF-8 BOM.
- When you use an exported file to import products:
  - Create an import file and add the information about the new products to the file.

#### NOTE

Do not change the name and sequence of table headers in the template. For details about the table header names, see [Description of Product Import Files](#).

- Open the import file and check whether any field value starting with **Base64:** exists. If there is no field value starting with **Base64:**, the file is exported from the instance of an earlier version.

In this case, rename the file by adding **-none-base64-prefix** to the end of the file name. For example, if the original file name is **import.csv**, change it to **import-none-base64-prefix.csv**.
- If you use Excel to edit the file to be imported, perform the following operations to save the file:
  - a. Save the Excel file as a CSV file by selecting **Comma Delimited files (\*.csv)**.
  - b. Use a text editor to open the CSV file saved in [a](#). Save the file in UTF-8 with BOM.

## Importing Products

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Product Management**. On the **Products** tab page, click **Import**.
3. In the dialog box displayed, select a local product file and import it.
4. After the import is successful, you can view the imported products in the product list.

#### NOTE

If product import fails, wait for 5 minutes before re-import.

## Exporting Products

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Product Management**.
3. Export specified or all products.
  - Exporting specified products: Select the products to be exported and click **Export** to export the product file to the local PC.



- Exporting all products: Click **Export All** to export all the products to the local PC.

## Description of Product Import Files

**Table 9-12** describes the requirements of a product import file.

**Table 9-12** Product import file requirements

| Column          | Description        | Remarks                                                                                           |
|-----------------|--------------------|---------------------------------------------------------------------------------------------------|
| ProductName     | Product name.      | Mandatory. The complexity requirements are the same as those for product creation on the console. |
| ProductSerial   | Product SN.        | Mandatory.                                                                                        |
| ManufacturerId  | Manufacturer ID.   | Mandatory. The complexity requirements are the same as those for product creation on the console. |
| ManufactureName | Manufacturer name. | Mandatory. The complexity requirements are the same as those for product creation on the console. |
| Model           | Product model.     | Mandatory. The complexity requirements are the same as those for product creation on the console. |
| ProductType     | Product type.      | Mandatory. The options are <b>0</b> (common product) and <b>1</b> (gateway product).              |
| ProtocolType    | Protocol type.     | Mandatory. The options are <b>MQTT</b> , <b>MODBUS</b> , and <b>OPCUA</b> .                       |
| Username        | Username.          | Optional.                                                                                         |
| Password        | Password.          | Optional.                                                                                         |
| appName         | Application name.  | Mandatory. The application name should already exist.                                             |
| DeviceType      | Device type.       | Mandatory. The default value is <b>Default</b> .                                                  |

| Column     | Description  | Remarks                                                          |
|------------|--------------|------------------------------------------------------------------|
| DataFormat | Data format. | Mandatory. The default value is <b>JSON</b> .                    |
| Services   | Thing model. | Optional. You are advised to create thing models on the console. |

## 9.5.3 Creating a Product Template

### Overview

You can save service capabilities of a product as a product template. When creating a product, you can select and inherit the service capabilities of the product template to quickly create the product.

### Creating a Product Template

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Product Management**. On the page displayed, click **Create Product Template** in the upper right corner.
3. In the **Create Product Template** dialog box, set product template parameters and click **OK**.

**Table 9-13** Parameters for creating a product template

| Parameter   | Description                                                                                                                                                                                                                  |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Name        | Enter a product template name. It is recommended that you enter a name based on naming rules to facilitate search.<br><b>NOTE</b><br>The name of each product template created in instances under the same tenant is unique. |
| Description | Enter the descriptive information of the product template.                                                                                                                                                                   |
| Status      | Specify whether to enable the product template (enabled by default). When disabled, the product template cannot add services and cannot be selected during product creation.                                                 |

### Adding a Thing Model Service to a Product Template

1. Log in to the ROMA Connect console, choose **LINK > Product Management**, and click the **Product Templates** tab.
2. Click the product template name to go to the product template details page.

3. Add a thing model service to a product template.
  - a. On the **Thing Model** tab page, click **Create**.
  - b. In the **Create Thing Model Service** dialog box, set thing model service parameters and click **OK**.

**Table 9-14** Parameters for creating a thing model service

| Parameter   | Description                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------------------|
| Name        | Enter a thing model service name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Status      | This parameter specifies whether to enable the thing model service. The thing model service is enabled by default.    |
| Description | Enter a brief description of the thing model service.                                                                 |

4. Add attributes for the thing model service.
  - a. On the **Thing Model** tab page, select the thing model service for which you want to add attributes.
  - b. On the **Attributes** tab page on the right, click **Create**.
  - c. In the **Create Attribute** dialog box, set attribute parameters and click **OK**.

**Table 9-15** Parameters for creating an attribute

| Parameter      | Description                                                                                                  |
|----------------|--------------------------------------------------------------------------------------------------------------|
| Attribute Name | Enter an attribute name. It is recommended that you enter a name based on naming rules to facilitate search. |


| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Data Type   | <p>Select the data type of the attribute.</p> <ul style="list-style-type: none"><li>• <b>Int</b>: integer. If you select <b>Int</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, and <b>Unit</b>.</li><li>• <b>Number</b>: number. If you select <b>Number</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, and <b>Unit</b>.</li><li>• <b>String</b>: a character string. If you select <b>String</b>, you also need to specify the maximum data length and enumerated values.</li><li>• <b>Bool</b>: Boolean. If you select <b>Bool</b>, you also need to specify <b>0</b> and <b>1</b>.</li><li>• <b>DateTime</b>: date. If you select <b>DateTime</b>, you also need to specify the maximum data length.</li><li>• <b>JsonObject</b>: JSON object. If you select <b>JsonObject</b>, you also need to specify the maximum data length.</li><li>• <b>Array</b>: array.</li></ul> <p>The following uses the <b>Int</b> type as an example:<br/>For example, when defining the temperature attribute for thermometers, set <b>Type</b> to <b>Int</b>, <b>Min. Value</b> to <b>0</b>, <b>Max. Value</b> to <b>100</b>, <b>Step</b> to <b>2</b>, and <b>Unit</b> to <b>°C</b>. This indicates that the device reports the temperature value (for example, 0°C, 2°C, 4°C, 6°C and 8°C) each time the temperature changes by two degrees.</p> |
| Mandatory   | This parameter specifies whether a device must report this attribute. This parameter is enabled by default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Description | Enter a description of the attribute.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

5. Add commands for the thing model service.
  - a. On the **Thing Model** tab page, select the thing model service for which you want to add commands.
  - b. On the **Commands** tab page, click **Create**.
  - c. In the **Create Command** dialog box, set command parameters and click **OK**.

**Table 9-16** Parameters for creating a command

| Parameter    | Description                                                                                               |
|--------------|-----------------------------------------------------------------------------------------------------------|
| Command Name | Enter a command name. It is recommended that you enter a name based on naming rules to facilitate search. |

| Parameter   | Description                               |
|-------------|-------------------------------------------|
| Description | Enter a brief description of the command. |

- d. Locate the created command in the command list and click  next to the command name to expand the command field list.

Click **Delivery Command Fields** or **Response Command Fields** on the right to view the delivery command fields and response command fields of the command.

- e. Click **Create Field**.
- f. In the **Create Field** dialog box, set command field parameters and click **OK**.

**Table 9-17** Parameters for creating a command field

| Parameter   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field Name  | Enter a field name. It is recommended that you enter a name based on naming rules to facilitate search.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Data Type   | Select the data type of the command field. <ul style="list-style-type: none"> <li>• <b>Int</b>: integer. If you select <b>Int</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, and <b>Unit</b>.</li> <li>• <b>Number</b>: number. If you select <b>Number</b>, you also need to specify <b>Min. Value</b>, <b>Max. Value</b>, and <b>Unit</b>.</li> <li>• <b>String</b>: a character string. If you select <b>String</b>, you also need to specify the maximum data length and enumerated values.</li> <li>• <b>Bool</b>: Boolean. If you select <b>Bool</b>, you also need to specify <b>0</b> and <b>1</b>.</li> <li>• <b>DateTime</b>: date. If you select <b>DateTime</b>, you also need to specify the maximum data length.</li> <li>• <b>JsonObject</b>: JSON object. If you select <b>JsonObject</b>, you also need to specify the maximum data length.</li> <li>• <b>Array</b>: array.</li> </ul> |
| Mandatory   | This parameter specifies whether the field must be carried in the delivery command. By default, the field must be carried.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Description | Enter the descriptive information of the command field.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

## 9.6 Device Management

## 9.6.1 Viewing Devices


### Overview

After a device is created, you can view the device password and details, force the device to go offline, and edit device information.

### Functions

After a device is created, you can view or operate the device by referring to the following table.

**Table 9-18** Functions

| Function                 | Description                                                                                                                                                                                                                                  |
|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Viewing device passwords | A device password can be used as a credential for accessing a device.<br><br>Click  on the right of the <b>Password</b> column to view the device password. |
| Editing a device         | Modify information of a created device, except the product name and device ID.                                                                                                                                                               |
| More > Force Offline     | Forcibly bring online devices offline.<br><b>NOTE</b><br>This function is applicable only to the online devices that use the MQTT protocol.                                                                                                  |

| Function                                                                                                                                                          | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>More &gt; Deliver Command</p> <p><b>NOTE</b><br/>Commands can be delivered only when the protocol type of the product to which the device belongs is MQTT.</p> | <p>LINK can effectively manage devices. The product thing model of a device defines the commands that can be delivered to the device. An application can deliver commands to a single device through LINK to remotely control the device. The command delivery configuration is as follows:</p> <ul style="list-style-type: none"> <li>• Select a thing model service that is created in the product to which the device belongs. If no thing model service is available, click <b>Create</b> on the right to create one.</li> <li>• Determine whether to enable <b>Synchronous Communication</b>. If this parameter is enabled, a success response is returned after the entire process is complete or a failure response is returned when the timeout period expires. If this parameter is disabled, LINK returns a response immediately after the command is delivered, regardless of the subsequent process. If the device is offline when the command is delivered, a success response is also returned. After the device goes online, LINK forwards the cached command to the device.</li> <li>• Select a command under the thing model service of the product.</li> <li>• (Optional) Set the delivered command field. This field is displayed based on the field configured in the command. If the field is not configured in the command, it is not displayed. For example, if the delivered field is temperature, a text box is provided to enter the temperature value, such as 37°C.</li> </ul> <p><b>NOTE</b><br/>An application can also call open northbound APIs of LINK to deliver commands to a single device or a batch of devices.</p> |
| More > Reset Password                                                                                                                                             | To change a password, click <b>Reset Password</b> in the <b>Operation</b> column of a device.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| More > Delete                                                                                                                                                     | Delete a device.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |

The content on the device details page varies depending on the product protocol type selected during device creation. For details, see [Table 9-19](#).

**Table 9-19** Description of the device details page

| Proto col Type | Tab Page          | Description                                                                                     |
|----------------|-------------------|-------------------------------------------------------------------------------------------------|
| MQTT           | Basic Information | You can view basic information, including the device ID, online status, and connection address. |

| Protocol Type | Tab Page          | Description                                                                                                                                                                                                               |
|---------------|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|               | Topics            | The system automatically generates default topics, that is, basic topics, by product ID and device ID. Custom topics are also supported. For details, see <a href="#">(Optional) Adding a Custom Topic for a Device</a> . |
|               | Device Shadows    | A device shadow is used to store the reported device status and values, which can be viewed in lists or JSON.                                                                                                             |
| Modbus        | Basic Information | View the basic device information, including the device ID, online status, connection address, edge connection configuration, and downloaded EdgeShell.                                                                   |
|               | Topics            | The system automatically generates default topics, that is, basic topics, by product ID and device ID. You can also customize topics as required. The procedure is the same as that when MQTT is used.                    |
|               | Edge Details      | View the resource status of the host where the device is located and the online and plug-in status.                                                                                                                       |



| Proto<br>col<br>Type | Tab Page             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | Modbus Configuration | <p>Create a Modbus connection point and policy as follows:</p> <ol style="list-style-type: none"> <li>1. Click the device name to go to the device details page. Click the <b>Modbus Configuration</b> tab.</li> <li>2. Click <b>Create Connection Point</b> and configure connection point information. <ul style="list-style-type: none"> <li>• Specify the connection point name in <b>Name</b>.</li> <li>• Select the connection point type. <b>TCP/IP</b> and <b>RS232</b> are supported.</li> <li>• Enter the timeout interval of a configuration request, in milliseconds.</li> <li>• Enter an IP address of the connection point.</li> <li>• Enter an IP port of the connection point.</li> </ul> </li> <li>3. Click <b>OK</b>.</li> <li>4. After the connection point is created, click the drop-down list button and click <b>Create Policy</b>. <ul style="list-style-type: none"> <li>• Enter a policy name.</li> <li>• Enter a value greater than 200 as the collection interval. The maximum polling interval is 86400000 (one day), in the unit of ms.</li> <li>• Enter a value ranging from 0 to 255 to as the slave site ID.</li> <li>• Enter a value ranging from 0 to 65535 as the start address.</li> <li>• Set the length of the data to be read. The value ranges from 1 to 65535.</li> <li>• Enter a function code. Currently, only <b>01</b>, <b>02</b>, <b>03</b>, and <b>04</b> are supported.</li> <li>• Click <b>Add Point</b>, configure the point mapping, and enter the attribute name of the point to be mapped. For example, if the attribute of collection point 0 is set to temperature, the 0-bit register value (temperature) is reported. This parameter is optional.</li> </ul> </li> <li>5. Click <b>OK</b>.</li> </ol> <p>After the policy is created, connect to a device, enable the plug-in, and click <b>Deliver Policy</b> to deliver the policy to the device.</p> |
|                      | Plug-in Deployment   | <p>Before using this function, you need to install the EdgeShell client and connect it to LINK.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

| Protocol Type | Tab Page             | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|---------------|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OPC UA        | Basic Information    | View the basic device information, including the device ID, online status, connection address, and downloaded EdgeShell.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|               | Topics               | The system automatically generates default topics, that is, basic topics, by product ID and device ID. You can also customize topics as required. The procedure is the same as that when MQTT is used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|               | Edge Details         | You can view the resource status of the host where the device is located, whether the device is online, and the plug-in status.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|               | OPC UA Configuration | <p>Create an OPC UA connection point and node.</p> <ol style="list-style-type: none"> <li>Click the device name to access the device details page. Click the <b>OPC UA Configuration</b> tab.</li> <li>Click <b>Create Connection Point</b> and configure connection point information. <ul style="list-style-type: none"> <li>Enter the name of the OPC UA connection point.</li> <li>Enter the TCP connection address, which must start with <b>opc.tcp:</b> and can contain only letters, digits, colons (:), and slashes (/). Example: <code>opc.tcp://127.0.0.1:34561/DataTransferServer</code></li> <li>Enter a value greater than 200 as the collection interval. The maximum polling interval is 86400000 (one day), in the unit of ms.</li> <li>Determine whether to enable regular collection. If this parameter is set to <b>Yes</b>, the collection start time is a multiple of the collection interval.</li> <li>Set a security policy.</li> <li>Select the security mode. If you select <b>User Authentication</b>, you also need to specify <b>Username</b> and <b>Password</b>.</li> <li>Enter the timeout interval of a configuration request, in milliseconds.</li> <li>Select a topic for the device to publish messages.</li> </ul> </li> <li>Click <b>OK</b>.</li> <li>After the connection point is created, click the drop-down list button, and click <b>Create Node</b> to enter the node path.</li> <li>Click <b>OK</b>.</li> </ol> <p>After the node is created, connect to a device, enable the plug-in, and click <b>Deliver Policy</b> to deliver the policy.</p> |

| Protocol Type | Tab Page           | Description                                                                                  |
|---------------|--------------------|----------------------------------------------------------------------------------------------|
|               | Plug-in Deployment | Before using this function, you need to install the EdgeShell client and connect it to LINK. |

## 9.6.2 Importing and Exporting Devices

### Overview

ROMA Connect supports device import and export. You can export device as a file to the local host or import a local product file to ROMA Connect to migrate device information in batches.

### Prerequisites

- Before importing a device, ensure that the integration application to which the device belongs has been created. Otherwise, [create the integration application](#) in advance.
- Before importing a device, ensure that the product to which the device belongs has been created. Otherwise, [create the product](#) in advance.
- Before importing a device, check whether a duplicate device name exists in the instance to which the device is to be imported. If there are duplicate device names, the device will fail to be imported.
- Ensure that the quota of devices meets the requirements before importing the devices.
- The files to be imported must be in CSV format and have been encoded using UTF-8 BOM.
- When you use an exported file to import devices:
  - Create an import file and add the information about the new devices to the file.

#### NOTE

Do not change the name and sequence of table headers in the template. For details about the table header names, see [Description of Device Import Files](#).

- Open the import file and check whether any field value starting with **Base64:** exists. If there is no field value starting with **Base64:**, the file is exported from the instance of an earlier version.

In this case, rename the file by adding **-none-base64-prefix** to the end of the file name. For example, if the original file name is **import.csv**, change it to **import-none-base64-prefix.csv**.
- If you use Excel to edit the file to be imported, perform the following operations to save the file:
  - a. Save the Excel file as a CSV file by selecting **Comma Delimited files (\*.csv)**.

- b. Use a text editor to open the CSV file saved in [a](#). Save the file in UTF-8 with BOM.

## Importing Devices

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **LINK > Device Management**. On the **Devices** tab page, click **Import Device**.
3. In the dialog box displayed, select a local device file and import it.
4. After the import is successful, you can view the imported devices in the device list.

## Exporting Devices

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **LINK > Device Management**.
3. Export devices.
  - Exporting specified products: Select the devices to be exported and click **Export** to export the device file to the local PC.
  - Exporting all devices: Click **Export All** to export all the devices to the local PC.

## Description of Device Import Files

[Table 9-20](#) describes the requirements of a device import file.

**Table 9-20** Device import file requirements

| Column             | Description                                                | Remarks                                                                                          |
|--------------------|------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| DeviceName         | Device name.                                               | Mandatory. The complexity requirements are the same as those for device creation on the console. |
| ProductSerial      | SN of the product to which the device belongs.             | Mandatory. The product SN should already exist.                                                  |
| ParentDeviceClient | Client ID of the gateway device to which a device belongs. | Optional. The gateway device should already exist.                                               |
| ClientID           | Client ID.                                                 | Optional.                                                                                        |

| Column           | Description         | Remarks                                                                                                                                                                                                                                                                                                                              |
|------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DeviceIdentifier | Device ID.          | Mandatory. The complexity requirements are the same as those for device creation on the console.                                                                                                                                                                                                                                     |
| Username         | Username.           | Optional.                                                                                                                                                                                                                                                                                                                            |
| Password         | Password.           | Optional. The complexity requirements are the same as those for device creation on the console.                                                                                                                                                                                                                                      |
| DeviceType       | Device type.        | Mandatory. The value depends on the protocol type of the product: <ul style="list-style-type: none"><li>• MQTT: <b>COMMON</b></li><li>• Modbus: <b>MODBUS</b></li><li>• OPC UA: <b>OPCUA</b></li></ul>                                                                                                                               |
| Remark           | Device description. | Optional.                                                                                                                                                                                                                                                                                                                            |
| customTopic      | Custom topic.       | Optional. You are advised to create custom topics on the console. <ul style="list-style-type: none"><li>• The topic format of publish permissions is product_ID/out/device_ID/topic_name.</li><li>• The topic format of subscribe permissions is product_ID/in/device_ID/topic_name.</li></ul> Separate two topics with a comma (,). |
| Enable           | Device status.      | Mandatory. The options are <b>0</b> (enabled) and <b>1</b> (disabled).                                                                                                                                                                                                                                                               |

## 9.6.3 Creating a Device Group

### Overview

If there are a large number of devices, manage these devices by creating groups, which helps you quickly search for your desired devices.

## Prerequisites

Each device group must belong to an integration application. Before creating a device group, ensure that an integration application is available. Otherwise, [create an integration application](#) first.

## Creating a Root Group

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Device Management**. On the **Device Groups** tab page, click **Create Root Group**.
3. In the **Create Root Group** dialog box, set group parameters and click **OK**.


**Table 9-21** Parameters for creating a root group

| Parameter               | Description                                                                                                    |
|-------------------------|----------------------------------------------------------------------------------------------------------------|
| Group Name              | Enter a device group name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Group Description       | Enter a brief description of the device group.                                                                 |
| Integration Application | Select the integration application to which the device group belongs.                                          |

## Adding a Subgroup

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Device Management**. On the **Device Groups** tab page, click a device group for which you want to add a subgroup.

The newly added subgroup and its devices must belong to the same application.

3. Click  on the right of the device group name.
4. In the **Create Group** dialog box, set subgroup parameters and click **OK**.

**Table 9-22** Parameters for creating a subgroup

| Parameter         | Description                                                                                                |
|-------------------|------------------------------------------------------------------------------------------------------------|
| Parent Group      | Parent group to which a subgroup belongs.                                                                  |
| Group Name        | Enter a subgroup name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Group Description | Enter a brief description of the device group.                                                             |

## Adding a Device to a Device Group

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Device Management**. On the **Device Groups** tab page, click a device group for which you want to add a device.
3. Click **Add Device to Group**.
4. In the **Add Device to Group** dialog box, select the devices to be added and click **OK**.

## 9.7 Rule Engine

### 9.7.1 Configuring Data Forwarding Rules

#### Overview

A rule engine can subscribe to device topics, obtain data reported by devices, and send parsed data to other cloud services for use. For example, a user can define a rule that requires a device to report its temperature every hour. If the device temperature exceeds the normal range, the device will be shut down and an alarm notification will be sent to the user. LINK transmits the collected data to the big data analysis platform to evaluate the risks of other devices.

#### Prerequisites

- Each data forwarding rule must belong to an integration application. Before creating a rule, ensure that an integration application is available. Otherwise, [create an integration application](#) first.
- To use a rule engine to forward data to DIS, the user must be assigned the **DIS Administrator** role.

#### Creating a Rule

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** of an instance.
2. In the navigation pane on the left, choose **LINK > Rule Engine**. On the page displayed, click **Create Rule** in the upper right corner.
3. In the **Create Rule** dialog box, set rule parameters and click **OK**.

**Table 9-23** Parameters for creating a rule

| Parameter               | Description                                                                                            |
|-------------------------|--------------------------------------------------------------------------------------------------------|
| Rule Name               | Enter a rule name. It is recommended that you enter a name based on naming rules to facilitate search. |
| Integration Application | Select the integration application to which the rule belongs.                                          |

| Parameter   | Description                                                                                                                          |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------|
| Description | Enter a brief description of the rule.                                                                                               |
| Status      | This parameter specifies whether to enable the rule. The rule is enabled by default. The rule takes effect only after being enabled. |

4. After the rule is created, click its name in the rule list to access the rule details page.
5. Configure a source for the rule.
  - a. Click **Create Source** in the **Source** area.
  - b. Configure source parameters and click **Save**.

**Table 9-24** Parameters for creating a source

| Parameter           | Description                                                                                                                                                                                                                                                                                                   |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Product Name        | Select the product to which a device belongs.                                                                                                                                                                                                                                                                 |
| Device Name         | Select the device that needs to forward data. You can select a specified device or all devices.                                                                                                                                                                                                               |
| Topic Name          | Select the topic used by the device to report messages.                                                                                                                                                                                                                                                       |
| Topic Level         | Select the topic level. The value is automatically adapted based on the value of <b>Device Name</b> . If you do not specify <b>Device Name</b> , <b>Topic Level</b> is <b>Product</b> . If you select a specific device from the <b>Device Name</b> drop-down list box, <b>Topic Level</b> is <b>Device</b> . |
| Base64 Encoding     | This parameter specifies whether Base64 encoding is performed on the forwarded device data.                                                                                                                                                                                                                   |
| Include Device Data | This parameter specifies whether the forwarded device information contains device data.                                                                                                                                                                                                                       |

6. (Optional) Configure data parsing to filter the data to be forwarded. For details about the SQL parsing configuration, see [SQL Parsing](#).

**NOTE**

After data parsing is applied, the **Base64 Encoding** and **Include Device Data** parameters do not take effect.

7. Configure a destination for the rule.
  - a. Click **Create Destination** in the **Destination** area.
  - b. Configure destination parameters and click **Save**.



**Table 9-25** Parameters for creating a destination

| Destination Type | Parameter          | Description                                                                                                                                                                                                                                              |
|------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ROMA MQS         | Connection Address | Select the connection address of ROMA MQS.                                                                                                                                                                                                               |
|                  | Topic Name         | Select the name of the topic to which data is to be forwarded.                                                                                                                                                                                           |
|                  | Username           | This parameter is available only if <b>MQS SASL_SSL</b> is enabled for the ROMA Connect instance.<br>Enter the key of the integration application to which the topic defined in <b>Topic Name</b> belongs.                                               |
|                  | Password           | This parameter is available only if <b>MQS SASL_SSL</b> is enabled for the ROMA Connect instance.<br>Enter the secret of the integration application to which the topic defined in <b>Topic Name</b> belongs.                                            |
| DMS for Kafka    | Connection Address | Select the connection address of DMS for Kafka from the drop-down list box.                                                                                                                                                                              |
|                  | Topic Name         | Select the name of the topic to which data is to be forwarded.                                                                                                                                                                                           |
|                  | Username           | This parameter is available only if <b>MQS SASL_SSL</b> is enabled for the ROMA Connect instance.<br>Enter the key of the integration application to which the topic defined in <b>Topic Name</b> belongs.                                               |
|                  | Password           | This parameter is available only if <b>MQS SASL_SSL</b> is enabled for the ROMA Connect instance.<br>Enter the secret of the integration application to which the topic defined in <b>Topic Name</b> belongs.                                            |
| DIS              | Stream             | Select the name of the DIS stream to which data is to be forwarded. A DIS stream is a logical unit created by tenants to distinguish real-time data of different tenants. When DIS is used to send or receive data, you need to specify the stream name. |

| Destination Type | Parameter | Description                                                                                                                                           |
|------------------|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
|                  | Service   | Select an entrusted service. The entrusted service allows a user to create an entrustment on IAM and grant ROMA Connect the permission to access DIS. |
| Device topic     | Product   | Select the product to which a device belongs.                                                                                                         |
|                  | Device    | Select the name of the device to which data is to be forwarded.                                                                                       |
|                  | Topic     | Select the name of the topic to which data is to be forwarded.                                                                                        |

## SQL Parsing

### Concepts

After a device is connected to ROMA Connect, the device encapsulates data into a message in JSON format and sends the message to ROMA Connect. JSON contains keys and values. A rule can be viewed as an SQL statement and JSON as a table. A key indicates a column header of the table, and its value indicates the corresponding key value. Device messages are filtered using the SQL statement and then sent to other services.

For example, a temperature sensor is used to control the temperature of devices. It can collect the device type, ambient temperature, ambient humidity, and current time. The format and content of the reported information is shown in the example.

```
{
  "device": "camera",
  "temperature": 30,
  "humidity": 65,
  "time": "xxx,xxx"
}
```

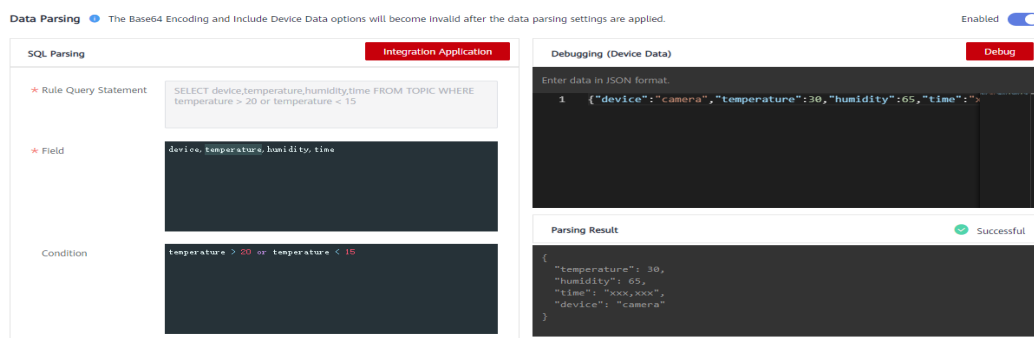
If you want to make a rule that requires an alarm message to be sent when the temperature is higher than 20 degrees Celsius or lower than 15 degrees Celsius, enter the following SQL statement: If the preceding conditions are met, LINK reports the device type, absolute value of the device temperature, device humidity, and time for further processing.

```
SELECT
  device, abs(temperature),
  humidity,
  TIME
FROM
  mcxeSR187154/OUT/test
WHERE
  temperature > 20
OR
  temperature < 15
```

**NOTE**

In the **FROM** statement, **mcxeSR187154/out/test** indicates that the rule engine accepts only messages from the device named **test**. Your device may be different from that in the example. Modify the device information based on the site requirements.

This rule is triggered when the temperature in reported data is greater than 20°C or less than 15°C. In addition, the temperature, device name, and location in the data will be parsed for further processing. [Figure 9-4](#) shows the data parsing result.

**Figure 9-4** Data parsing result**Usage Description**

An SQL statement consists of three parts: **SELECT**, **FROM**, and **WHERE**. Data in JSON format is classified into two types: constant data with single or double quotation marks and variable data without any quotation marks.

The field in the **SELECT** statement is the value of the key in the JSON message. Built-in SQL functions are supported. For details about how to use other SQL functions, see [Table 9-27](#). In addition, the SELECT statement supports not only the combination of asterisks (\*) and functions, but also arrays and JSON with nested values. For example, you can use `a.color` to obtain `"color": "red"` in `{"a": {"temperature": 29, "color": "red"}}`. Note that the fields without quotation marks are variables, and the fields with single quotation marks and double quotation marks are constants.

The temperature can be a positive number, 0, or negative number. To facilitate management, **abs(temperature)** in the preceding example uses the `abs()` function to output the absolute value of the temperature.

The **FROM** statement contains the device name. You can specify a single device or all devices of a product to report messages. *Product name\_out\_Device name* indicates a single device. After the rule is executed, it applies only to this device. *Product name\_out\_+* can match all devices under this product because the plus sign (+) represents all items in this level. After the rule is executed, it applies to all devices under the product.

The **WHERE** statement contains condition expressions and is used to filter fields and messages that meet conditions. In the preceding example, **WHERE temperature > 20 or temperature < 15** is the filter criterion. Messages will be filtered out only when the temperature is higher than 20°C or lower than 15°C. For details about the condition expressions supported by the **WHERE** statement, see [Table 9-26](#).

**Table 9-26** Condition expression

| Operator                                        | Description                                               | Example                                                                                                                   |
|-------------------------------------------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| =                                               | Equal                                                     | color = 'red'                                                                                                             |
| <>                                              | Not equal to                                              | color <>'red'                                                                                                             |
| and                                             | And                                                       | color = 'red' and switch = 'on'                                                                                           |
| or                                              | Or                                                        | color = 'red' or switch = 'on'                                                                                            |
| ( )                                             | Represent a whole.                                        | a>1 and (b<1 or b>5). The judgment of the OR logic is executed first, and then the judgment of the AND logic is executed. |
| in                                              | Only enumeration is supported. Subquery is not supported. | where a in(1,2,3). The example of <b>where a in(select xxx)</b> is not supported.                                         |
| +                                               | Addition                                                  | a in (3,2,3+8)                                                                                                            |
| -                                               | Subtraction                                               | 13 - 2                                                                                                                    |
| /                                               | Divide                                                    | 25 / 5                                                                                                                    |
| *                                               | Multiple                                                  | 2 * 8                                                                                                                     |
| %                                               | Obtains the remainder.                                    | 10 % 2                                                                                                                    |
| <                                               | Less than                                                 | 1 < 3                                                                                                                     |
| <=                                              | Less than or equal to                                     | 1 <= 3                                                                                                                    |
| >                                               | Greater than                                              | 8 > 3                                                                                                                     |
| >=                                              | Greater than or equal to                                  | 8 >= 3                                                                                                                    |
| CASE ...<br>WHEN ...<br>THEN ...<br>ELSE ...END | Case expression (Nested expression is not supported.)     | <b>case a when 3 then 'hello' when 4 then 'bye' end</b> FROM item WHERE a >= b+c"                                         |

A rule engine also provides a variety of functions that you can use when you write SQL statements to diversify your data processing. You can use functions in SQL statements to obtain or process data. For example, "**SELECT** service, abs(temperature)" uses the abs(number) function. For details about the condition expression, see [Table 9-27](#).

**Table 9-27** SQL functions

| Function                                | Description                                                                                                                                                                                |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| abs(number)                             | Returns the absolute value.                                                                                                                                                                |
| sin(n)                                  | Returns the arc sine of the n value.                                                                                                                                                       |
| cos(number)                             | Returns the arc sine of the number value.                                                                                                                                                  |
| asin(number)                            | Returns the arc sine of the number value.                                                                                                                                                  |
| sinh(n)                                 | Returns the hyperbolic cosine of the n value.                                                                                                                                              |
| cosh(number)                            | Returns the hyperbolic cosine of the number value.                                                                                                                                         |
| tan(n)                                  | Returns the tangent of the n value.                                                                                                                                                        |
| tanh(n)                                 | Returns the hyperbolic tangent of the n value.                                                                                                                                             |
| lower(string)                           | Returns the lowercase letters.                                                                                                                                                             |
| upper(string)                           | Returns the uppercase letters.                                                                                                                                                             |
| power(n,m)                              | Return the value of <b>n</b> raised to the power of <b>m</b> .                                                                                                                             |
| rand()                                  | Returns a random number ranging from 0 to 1.                                                                                                                                               |
| mod(n, m)                               | Returns the remainder of $n\%m$ .                                                                                                                                                          |
| log(n, m)                               | Returns the natural logarithm. If the value of m does not need to be transferred, log(n) is returned.                                                                                      |
| exp(number)                             | Returns the specified power of a number.                                                                                                                                                   |
| floor(number)                           | Returns the closest integer, whose value is less than or equal to the floating point number.                                                                                               |
| concat(string1, string2)                | String concatenation. For example, concat(field,a) outputs <b>fielda</b> .                                                                                                                 |
| replace(source, substring, replacement) | Replaces the value of a target column. For example, replace(field,'iel','oo') outputs <b>food</b> .                                                                                        |
| topic()                                 | Returns the information about the entire topic. For example, Topic: /abcdef/ghi. Use the topic() function to return / <b>abcdef/ghi</b> .                                                  |
| endswith(input, suffix)                 | Checks whether the input value ends with the suffix.                                                                                                                                       |
| timestamp(format)                       | If no parameter is specified, the default timestamp is returned. If a parameter is specified, the timestamp in the specified format is returned. For example, timestamp() = 1553572557420. |

| Function       | Description                                                                                                                                                                                                              |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| timestampUtc() | Obtains the UTC time of the system. If no parameter is specified, the current system time in milliseconds is returned. If one parameter is specified, the parameter is used as the format parameter for time formatting. |
| serviceId()    | Returns the service ID corresponding to the message. The parameter in this function must be left unspecified.                                                                                                            |
| clientId()     | Obtains the client ID of the current topic. The parameter in this function must be left unspecified.                                                                                                                     |

## 9.7.2 Importing and Exporting Rules

### Overview

ROMA Connect supports rule import and export. You can export rules to a file to the local host or import a local rule file to ROMA Connect to migrate rule information in batches.

### Prerequisites

- Before importing a rule, ensure that the integration application to which the rule belongs has been created. Otherwise, [create an integration application](#) first.
- Before importing a rule, ensure that the product to which the rule belongs has been created. Otherwise, [create the product](#) in advance.
- Before importing a rule, ensure that the device to which the rule belongs has been created. Otherwise, [Registering a Device](#) in advance.
- Before importing a rule, check whether a duplicate rule name exists in the instance to which the rule is to be imported. If there are duplicate rule names, the rule will fail to be imported.
- Ensure that the quota of rules meets the requirements before importing the rules.
- The files to be imported must be in CSV format and have been encoded using UTF-8 BOM.
- When you use an exported file to import rules:
  - Create an import file and add the information about the new rules to the file.

#### NOTE

- Do not change the name and sequence of table headers in the template. For details about the table header names, see [Description of Rule Import Files](#).
- Open the import file and check whether any field value starting with **Base64:** exists. If there is no field value starting with **Base64:**, the file is exported from the instance of an earlier version.

In this case, rename the file by adding **-none-base64-prefix** to the end of the file name. For example, if the original file name is **import.csv**, change it to **import-none-base64-prefix.csv**.

- If you use Excel to edit the file to be imported, perform the following operations to save the file:
  - a. Save the file as a CSV file by selecting **Comma Delimited files (\*.csv)**.
  - b. Use a text editor to open the CSV file saved in **a**. Save the file in UTF-8 with BOM.

## Importing Rules

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Rule Engine**. On the **Rule Engine** page, click **Import**.
3. In the dialog box displayed, select a local rule file and import it.
4. After the import is successful, you can view the imported rules in the rule list.

## Exporting Rules

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Rule Engine**.
3. Export rules.
  - Exporting specified rules: Select the rules to be exported and click **Export** to export the rule file to the local PC.
  - Export all rules: Click **Export All**. The system will export all the rules to the local PC.

## Description of Rule Import Files

**Table 9-28** describes the requirements of a rule import file.

**Table 9-28** Rule import file requirements

| Column | Description       | Remarks                                                                                        |
|--------|-------------------|------------------------------------------------------------------------------------------------|
| Name   | Rule name.        | Mandatory. The complexity requirements are the same as those for rule creation on the console. |
| Remark | Rule description. | Optional.                                                                                      |
| Status | Status.           | Mandatory. The options are <b>0</b> (enabled) and <b>1</b> (disabled).                         |

| Column            | Description                     | Remarks                                                                |
|-------------------|---------------------------------|------------------------------------------------------------------------|
| DataParsingStatus | Data parsing status.            | Mandatory. The options are <b>0</b> (enabled) and <b>1</b> (disabled). |
| SqlField          | Query field.                    | Optional. You are advised to create query fields on the console.       |
| SqlWhere          | Query condition.                | Optional. You are advised to create query conditions on the console.   |
| AppName           | Application name.               | Mandatory. The application name should already exist.                  |
| Sources           | Data source configuration.      | Optional. You are advised to create data sources on the console.       |
| Destinations      | Data destination configuration. | Optional. You are advised to create data destinations on the console.  |

## 9.8 Subscription Management

### 9.8.1 Subscribing to Device Notifications

#### Overview

ROMA Connect provides the device change notification function. When the status of a subscribed device changes, for example, the device goes online, goes offline, or is deleted, ROMA Connect sends a message to the corresponding message integration topic to obtain the latest device status in real time.

#### Prerequisites

A topic is available for receiving messages, and the topic is in the same application as a device. Otherwise, [create a topic](#) first.

#### Procedure

1. Log in to the ROMA Connect console. On the **Instances** page, click **View Console** next to a specific instance.
2. In the navigation pane on the left, choose **LINK > Subscription Management**.



3. On the **Subscription Management** page, select the integration application to which the topic that needs to receive notifications belongs.
4. Under the notification type to be subscribed to, select a topic to receive notifications and enable the notification function.
  - After a notification function is enabled or disabled, it takes a maximum of 30 seconds to take effect.
  - A notification is triggered when any of the following operations is performed:
    - The device type is changed. For example, a change from a common device to a subdevice.
    - The device name is changed.
    - The device status is changed.
    - The device description is changed.
  - For details about the notification message example, see [Appendix: Packets of Subscribed Notification Messages](#).

## 9.8.2 Appendix: Packets of Subscribed Notification Messages

This section uses a common device as an example to describe sample MQS packets related to subscription management.

- Message attributes

```
{
  "deviceIdentifier":"iottest2", //Device identifier
  "eventTime":"1588843914884", //Message time
  "gatewayIdentifier":"null", //Gateway identifier
  "deviceId":"D114VKx7a", //Device ID
  "gatewayId":"null", //Gateway ID
  "TAGS":"null" //Message tag
}
```

### NOTE

- For a common device or gateway, set **gatewayId** and **gatewayIdentifier** to **null**.
  - For a subdevice, **gatewayId** and **gatewayIdentifier** are **deviceId** and **deviceIdentifier** of the gateway respectively.
- Message content

- a. Adding a device

```
{
  'deviceType':'Default', //Device type
  'instanceOid':'40', //Instance OID
  'manufacturerName':'iotcompro1', //Manufacturer name
  'productOid':'1', //Product OID
  'manufacturerId':'iotcompro1', //Manufacturer ID
  'remark':"", //Device description
  'deviceId':'D114VKx7a', //Device ID
  'deviceName':'iottest2', //Device name
  'productName':'iotcompro1', //Product name
  'notifyType':'deviceAdded', //Message type
  'deviceIdentifier':'iottest2', //Device identifier
  'deviceOid':'14', //Device OID
  'createdBy':'admin', //Creator
  'eventTime':'1588843914884', //Message time
  'model':'iotcompro1', //Product model
  'productType':'0', //Product type. 1: gateway product; 0: common product.
}
```

```
'status':'0', //Device status. 0: enabled; 1: disabled.  
'parentDeviceOid':'null' //Parent device OID  
}
```

b. Deleting a device

```
{  
  'notifyType':'deviceDeleted', //Message type  
  'deviceIdentifier':'iottest2', //Device identifier  
  'eventTime':'1588843914884', //Message time  
  'deviceId':'D114VKx7a', //Device ID  
}
```

c. Taking a device online

```
{  
  'notifyType':'deviceOnline', //Message type: device online  
  'deviceIdentifier':'iottest2', //Device identifier  
  'eventTime':'1588843914884', //Message time  
  'deviceId':'D114VKx7a', //Device ID  
}
```

d. Taking a device offline

```
{  
  'notifyType':'deviceDeleted', //Message type  
  'deviceIdentifier':'iottest2', //Device identifier  
  'eventTime':'1588843914884', //Message time  
  'deviceId':'D114VKx7a', //Device ID  
}
```

e. Changing the device type

```
{  
  'instanceOid':'40', //Instance OID  
  'productOid':'1', //Product OID  
  'remark':'', //Device description  
  'deviceId':'D114VKx7a', //Device ID  
  'deviceName':'iottest1', //Device name  
  'notifyType':'deviceChange', //Message type  
  'deviceIdentifier':'iottest1', //Device identifier  
  'deviceOid':'1', //Device OID  
  'createdBy':'admin', //Creator  
  'eventTime':'1588843914884', //Message time  
  'status':'0', //Device status. 0: enabled; 1: disabled.  
  'parentDeviceOid':'null' //Parent device OID  
}
```

# 10 Increasing Resource Quota

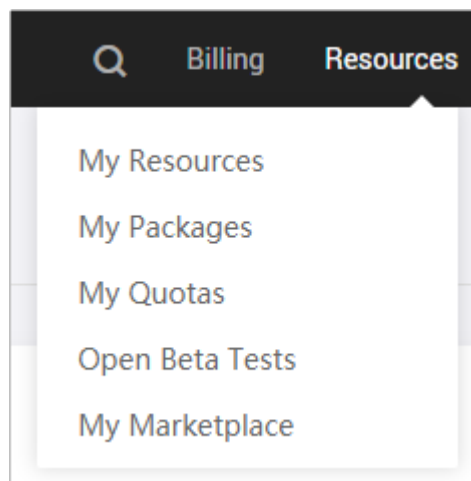
## Overview

To prevent resource abuse, ROMA Connect limits the number of resources that can be created, such as the number of instances, APIs, and devices. If the existing resource quota cannot meet your service requirements, you can apply for a higher quota.

## Procedure

**Step 1** In the upper right corner of the ROMA console, choose **Resources > My Quotas**.

**Figure 10-1** My quota



**Step 2** Click **Increase quota**.

**Step 3** On the page displayed, enter key information, select **I have read and agree to the Tenant Authorization Letter and Privacy Statement**, and click **Submit**.

**Table 10-1** Parameters for increasing the instance quota

| Parameter                | Description                                                                                                                                                               |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Region                   | Select the region where the target ROMA Connect service resides.                                                                                                          |
| Problem Description      | Describe your needs, for example, "Applying for ROMA Connect service instance quota".                                                                                     |
| Confidential Information | Retain the default value <b>Enter later</b> .                                                                                                                             |
| Contact Method           | Retain the default settings (both <b>Mobile</b> and <b>Email</b> selected).                                                                                               |
| Mobile Number            | Enter your mobile number.                                                                                                                                                 |
| Call Me at               | Select <b>Any Time</b> or <b>Set Time</b> as required. If <b>Set Time</b> is selected, you also need to set the contact time segment.                                     |
| Email Address            | Retain the default value, that is, the email address bound to the account. If your account is not bound to an email address, you need to manually enter an email address. |

**Step 4** After the service ticket is successfully submitted, contact customer service.

----End

# 11 Audit Logs

## Overview

Cloud Trace Service (CTS) can record operations related to ROMA Connect for query, audit, and backtracking.

After you enable CTS, the system starts to record ROMA Connect operations. CTS stores operation records from the last seven days.

## Enabling CTS

For details about how to enable CTS, see [Enabling CTS](#).

After CTS is enabled, if you want to view ROMA Connect operation events, see [Querying Real-Time Traces](#).

## ROMA Connect Operations Supported by CTS

**Table 11-1** ROMA Connect operations that can be recorded by CTS

| Operation                                  | Resource Type | Trace Name               |
|--------------------------------------------|---------------|--------------------------|
| Creating a ROMA Connect instance           | instance      | createRomaInstance       |
| Result of creating a ROMA Connect instance | instance      | createRomaInstanceResult |
| Deleting a ROMA Connect instance           | instance      | deleteRomaInstance       |
| Result of deleting a ROMA Connect instance | instance      | deleteRomaInstanceResult |
| Modifying a ROMA Connect instance          | instance      | updateRomaInstance       |

| Operation                                                        | Resource Type | Trace Name                      |
|------------------------------------------------------------------|---------------|---------------------------------|
| Modifying the name of a ROMA Connect instance                    | instance      | updateRomaInstanceName          |
| Modifying the description of a ROMA Connect instance             | instance      | updateRomaInstanceDesc          |
| Modifying the security group of a ROMA Connect instance          | instance      | updateRomaInstanceSecurityGroup |
| Modifying the maintenance time window of a ROMA Connect instance | instance      | updateRomaInstanceMaintain-Time |
| Modifying the EIP of a ROMA Connect instance                     | instance      | updateRomaInstanceEip           |
| Deleting instances that failed to be created                     | instance      | cleanFailedRomaInstance         |

## ROMA FDI Operations Supported by CTS

**Table 11-2** ROMA FDI operations that can be recorded by CTS

| Operation                            | Resource Type | Trace Name       |
|--------------------------------------|---------------|------------------|
| Creating a task                      | task          | createTask       |
| Editing a task                       | task          | updateTask       |
| Deleting a task                      | task          | deleteTask       |
| Starting or stopping a task          | task          | operateTask      |
| Manually starting or stopping a task | task          | handleTask       |
| Creating a scheduling plan           | task          | addDispatch      |
| Editing a scheduling plan            | task          | updateDispatch   |
| Connecting to a data source          | dataSource    | createDatasource |

| Operation                                                                               | Resource Type | Trace Name       |
|-----------------------------------------------------------------------------------------|---------------|------------------|
| Editing a data source                                                                   | dataSource    | updateDatasource |
| Deleting a data source                                                                  | dataSource    | deleteDatasource |
| Specifying an SQL Statement to query table field information of the plug-in data source | column        | checkSql         |

## ROMA APIC Operations Supported by CTS

**Table 11-3** ROMA APIC operations that can be recorded by CTS

| Operation                                        | Resource Type | Trace Name              |
|--------------------------------------------------|---------------|-------------------------|
| Creating an API group                            | ApiGroup      | createApiGroup          |
| Deleting an API group                            | ApiGroup      | deleteApiGroup          |
| Updating an API group                            | ApiGroup      | updateApiGroup          |
| Adding the access control permission             | ApiGroup      | deleteApiGroupLimit     |
| Deleting the access control permission           | ApiGroup      | updateApiGroupLimit     |
| Binding a custom domain name to an API group     | ApiGroup      | createDomainBinding     |
| Unbinding a custom domain name from an API group | ApiGroup      | relieveDomainBinding    |
| Adding a certificate to a custom domain name     | ApiGroup      | addDomainCertificate    |
| Deleting a certificate for a custom domain name  | ApiGroup      | deleteDomainCertificate |
| Creating an API                                  | Api           | createApi               |
| Deleting an API                                  | Api           | deleteApi               |
| Deleting APIs in batches                         | Api           | batchDeleteApi          |
| Modifying an API                                 | Api           | updateApi               |

| Operation                                         | Resource Type | Trace Name               |
|---------------------------------------------------|---------------|--------------------------|
| Publishing an API                                 | Api           | publishApi               |
| Taking an API offline                             | Api           | offlineApi               |
| Publishing APIs or unpublishing APIs in batches   | Api           | batchPublishOrOfflineApi |
| Switching API versions                            | Api           | switchApiVersion         |
| Taking an API offline by version                  | Api           | offlineApiByVersion      |
| Debugging an API                                  | Api           | debugApi                 |
| Creating an environment                           | Environment   | createEnvironment        |
| Deleting an environment                           | Environment   | deleteEnvironment        |
| Modifying an environment                          | Environment   | updateEnvironment        |
| Creating an environment variable                  | EnvVariable   | createEnvVariable        |
| Deleting an environment variable                  | EnvVariable   | deleteEnvVariable        |
| Creating an integration application               | App           | createApp                |
| Deleting an integration application               | App           | deleteApp                |
| Modifying an integrated application               | App           | updateApp                |
| Resetting a secret for an integration application | App           | resetAppSecret           |
| Authorization                                     | AppAuth       | grantAuth                |
| Canceling authorization                           | AppAuth       | relieveAuth              |
| Creating a signature key                          | Signature     | createSignature          |
| Deleting a signature key                          | Signature     | deleteSignature          |
| Modifying a signature key                         | Signature     | updateSignature          |



| Operation                                            | Resource Type    | Trace Name              |
|------------------------------------------------------|------------------|-------------------------|
| Binding a signature key                              | SignatureBinding | createSignatureBinding  |
| Unbinding a signature key                            | SignatureBinding | relieveSignatureBinding |
| Creating an access control policy                    | Acl              | createAcl               |
| Deleting an access control policy                    | Acl              | deleteAcl               |
| Deleting multiple access control policies            | Acl              | batchDeleteAcl          |
| Modifying an access control policy                   | Acl              | updateAcl               |
| Adding the value of the access control policy        | Acl              | addAclValue             |
| Deleting the value of the access control policy      | Acl              | deleteAclValue          |
| Binding an access control policy to an API           | AclBinding       | createAclBinding        |
| Unbinding an access control policy from an API       | AclBinding       | relieveAclBinding       |
| Unbinding multiple access control policies from APIs | AclBinding       | batchRelieveAclBinding  |
| Creating a request throttling policy                 | Throttle         | createThrottle          |
| Deleting a request throttling policy                 | Throttle         | deleteThrottle          |
| Deleting multiple request throttling policies        | Throttle         | batchDeleteThrottle     |
| Modifying a request throttling policy                | Throttle         | updateThrottle          |
| Binding a request throttling policy                  | ThrottleBinding  | createThrottleBinding   |

| Operation                                               | Resource Type   | Trace Name                   |
|---------------------------------------------------------|-----------------|------------------------------|
| Unbinding a request throttling policy                   | ThrottleBinding | relieveThrottleBinding       |
| Unbinding multiple request throttling policies          | ThrottleBinding | batchRelieveThrottleBinding  |
| Creating a special request throttling policy            | ThrottleSpecial | createSpecialThrottle        |
| Deleting a special request throttling policy            | ThrottleSpecial | deleteSpecialThrottle        |
| Modifying a special request throttling policy           | ThrottleSpecial | updateSpecialThrottle        |
| Creating a load balance channel                         | Vpc             | createVpc                    |
| Deleting a load balance channel                         | Vpc             | deleteVpc                    |
| Modifying a load balance channel                        | Vpc             | updateVpc                    |
| Adding a backend instance to a load balance channel     | Vpc             | addVpcMember                 |
| Deleting a backend instance from a load balance channel | Vpc             | deleteVpcMember              |
| Exporting an API                                        | Swagger         | swaggerExportApi             |
| Exporting multiple APIs                                 | Swagger         | swaggerExportApiList         |
| Exporting all APIs in an API group                      | Swagger         | swaggerExportApiByGroup      |
| Importing APIs to a new API group                       | Swagger         | swaggerImportApiToNewGroup   |
| Importing APIs to an existing API group                 | Swagger         | swaggerImportApiToExistGroup |
| Creating a custom authorizer                            | Authorizer      | createAuthorizer             |
| Deleting a custom authorizer                            | Authorizer      | deleteAuthorizer             |

| Operation                     | Resource Type | Trace Name       |
|-------------------------------|---------------|------------------|
| Modifying a custom authorizer | Authorizer    | updateAuthorizer |

## ROMA MQS Operations Supported by CTS

**Table 11-4** ROMA MQS operations that can be recorded by CTS

| Operation                                                                                       | Resource Type | Trace Name                     |
|-------------------------------------------------------------------------------------------------|---------------|--------------------------------|
| Creating a topic in a Kafka-based MQS instance (successful)                                     | kafka         | Kafka_create_topicSuccess      |
| Creating a topic in a Kafka-based MQS instance (failed)                                         | kafka         | Kafka_create_topicFailure      |
| Deleting a topic in a Kafka-based MQS instance (successful)                                     | kafka         | Kafka_delete_topicSuccess      |
| Deleting a topic in a Kafka-based MQS instance (failed)                                         | kafka         | Kafka_delete_topicsFailure     |
| Modifying a topic in a Kafka-based MQS instance (successful)                                    | kafka         | Kafka_alter_topicsSuccess      |
| Modifying a topic in a Kafka-based MQS instance (failed)                                        | kafka         | Kafka_alter_topicsFailure      |
| Configuring the integration application permissions for a Kafka-based MQS instance (successful) | kafka         | updateAppPoliciesSuccess       |
| Configuring the integration application permissions for a Kafka-based MQS instance (failed)     | kafka         | updateAppPoliciesFailure       |
| Creating a topic in a RocketMQ-based MQS instance (successful)                                  | rocketmq      | RocketMQ_Topic_CreationSuccess |

| Operation                                                                | Resource Type | Trace Name                     |
|--------------------------------------------------------------------------|---------------|--------------------------------|
| Creating a topic in a RocketMQ-based MQS instance (failed)               | rocketmq      | RocketMQ_Topic_CreationFailure |
| Modifying a topic in a RocketMQ-based MQS instance (successful)          | rocketmq      | RocketMQ_Topic_ModifySuccess   |
| Modifying a topic in a RocketMQ-based MQS instance (failed)              | rocketmq      | RocketMQ_Topic_ModifyFailure   |
| Deleting a topic in a RocketMQ-based MQS instance (successful)           | rocketmq      | RocketMQ_Topic_DeletionSuccess |
| Deleting a topic in a RocketMQ-based MQS instance (failed)               | rocketmq      | RocketMQ_Topic_DeletionFailure |
| Creating a consumer group in a RocketMQ-based MQS instance (successful)  | rocketmq      | RocketMQ_Create_GroupSuccess   |
| Creating a consumer group in a RocketMQ-based MQS instance (failed)      | rocketmq      | RocketMQ_Create_GroupFailure   |
| Modifying a consumer group in a RocketMQ-based MQS instance (successful) | rocketmq      | RocketMQ_Group_ModifySuccess   |
| Modifying a consumer group in a RocketMQ-based MQS instance (failed)     | rocketmq      | RocketMQ_Group_ModifyFailure   |
| Deleting a consumer group in a RocketMQ-based MQS instance (successful)  | rocketmq      | RocketMQ_Group_DeletionSuccess |
| Deleting a consumer group in a RocketMQ-based MQS instance (failed)      | rocketmq      | RocketMQ_Group_DeletionFailure |

## ROMA LINK Operations Supported by CTS

**Table 11-5** ROMA LINK operations that can be recorded by CTS

| Operation                             | Resource Type | Trace Name              |
|---------------------------------------|---------------|-------------------------|
| Creating a product                    | Product       | createProduct           |
| Updating a product                    | Product       | updateProduct           |
| Deleting a product                    | Product       | deleteProduct           |
| Uploading a product                   | Product       | uploadProducts          |
| Resetting a product password          | Product       | resetProductPassword    |
| Creating a product attribute          | Product       | addProductAttribute     |
| Updating a product attribute          | Product       | updateProductAttribute  |
| Deleting a product attribute          | Product       | removeProductAttribute  |
| Creating a product template           | Product       | createProductTemplate   |
| Updating a product template           | Product       | updateProductTemplate   |
| Deleting a product template           | Product       | deleteProductTemplate   |
| Creating a product template attribute | Product       | addTemplateAttribute    |
| Updating a product template attribute | Product       | updateTemplateAttribute |
| Deleting a product template attribute | Product       | removeTemplateAttribute |
| Creating a device                     | Device        | createDevice            |
| Updating a device                     | Device        | updateDevice            |
| Deleting a device                     | Device        | deleteDevice            |
| Uploading a device                    | Device        | uploadDevices           |
| Resetting a device password           | Device        | resetDevicePassword     |
| Creating a topic                      | Topic         | createTopic             |
| Updating a topic                      | Topic         | updateTopic             |

| Operation                   | Resource Type | Trace Name            |
|-----------------------------|---------------|-----------------------|
| Deleting a topic            | Topic         | deleteTopic           |
| Updating the device shadow  | Device        | updateDeviceShadow    |
| Creating a rule             | Rule          | createRule            |
| Updating a rule             | Rule          | updateRule            |
| Deleting a rule             | Rule          | deleteRule            |
| Adding a source rule        | Rule          | addRuleSource         |
| Deleting a source rule      | Rule          | removeRuleSource      |
| Adding a destination rule   | Rule          | addRuleDestination    |
| Deleting a destination rule | Rule          | removeRuleDestination |

## ROMA BFS Operations Supported by CTS

Table 11-6 ROMA BFS operations that can be recorded by CTS

| Operation                    | Resource Type | Trace Name             |
|------------------------------|---------------|------------------------|
| Creating a graph             | graph         | createGraph            |
| Updating the graph           | graph         | updateGraph            |
| Starting a graph             | graph         | batchStartGraphs       |
| Stopping a graph             | graph         | batchStopGraphs        |
| Deleting a graph             | graph         | batchDeleteGraphs      |
| Deleting an asset            | graph         | deleteAssets           |
| Exporting assets             | assets        | handleExportFlowAssets |
| Importing assets             | assets        | handleImportFlowAssets |
| Creating an integration task | integration   | createIntegration      |
| Updating an integration task | integration   | updateNewIntegration   |
| Deleting an integration task | integration   | deleteIntegration      |
| Buying a business flow       | purchase      | purchaseBfs            |

# 12 Monitoring Metrics

## Overview

Cloud Eye monitors the running status of cloud services and usage of each metric, and creates alarm rules for monitoring metrics.

After you enable ROMA Connect, Cloud Eye automatically associates with ROMA Connect monitoring metrics to help you understand the running status of ROMA Connect.

## Enabling Cloud Eye

Cloud Eye is enabled by default.

For details about how to view ROMA Connect monitoring metrics, see [Querying Cloud Service Monitoring Metrics](#).

Create an alarm rule to send an alarm notification when the monitoring data meets the specified conditions. For details, see [Creating an Alarm Rule](#).

## Metrics Supported by FDI

Table 12-1 Metrics supported by FDI

| Category | Metrics      | Description                                                                                                                                                                                                                           |
|----------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Instance | Active Tasks | Total number of running tasks in the instance. Configure this metric when you want to receive alarm notifications once an exception occurs. This metric is suitable for stable projects where there are few changes in task quantity. |

| Category | Metrics                                            | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|----------|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | Total Tasks                                        | Total number of FDI tasks in the instance, regardless of the running status.<br>Configure this metric when you want to receive alarm notifications once a task is mistakenly deleted. This metric is suitable for stable projects where there are few changes in task quantity or only few tasks need to be added or deleted.                                                                                                                                        |
|          | Data Size                                          | Total size of data written by all tasks in the instance in the last period (5 minutes).<br>Configure this metric when you want to receive alarm notifications once the maximum or minimum size of written data is exceeded.                                                                                                                                                                                                                                          |
|          | Data Records                                       | Total records of data written by all tasks in the instance in the last period (5 minutes).<br>Configure this metric when you want to receive alarm notifications once the maximum or minimum number of written data records is exceeded.                                                                                                                                                                                                                             |
|          | Successful Tasks                                   | Total number of successful tasks in the instance in the last period (5 minutes).                                                                                                                                                                                                                                                                                                                                                                                     |
|          | Failed Tasks                                       | Total number of failed tasks in the instance in the last period (5 minutes).                                                                                                                                                                                                                                                                                                                                                                                         |
| Task     | Failed Count                                       | Number of times a task fails to be executed in the last period (5 minutes).                                                                                                                                                                                                                                                                                                                                                                                          |
|          | Delay of Earliest Transaction Not Submitted by CDC | The difference between the time of the earliest transaction that is being processed but not submitted by the CDC composite task and the current time.<br>For example, for a MySQL task, this metric indicates the difference between the current system time and the time when the binary log is being read in the task. The value of this metric is consistent with that of <b>Real-time read monitoring at the read end</b> on the <b>View Log</b> page of a task. |



| Category | Metrics                                      | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------|----------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|          | Delay of Latest Transaction Submitted by CDC | <p>The interval between the time of the latest transaction that has been submitted by the CDC composite task and the current time. This metric applies only to Oracle tasks.</p> <p>It indicates the difference between the current time and the time of the latest transaction that has been CDC processed and successfully synchronized to the destination. Configure the delay threshold based on the actual project data volume. A value of greater than or equal to 3600 seconds (1 hour) is recommended.</p> |
|          | Oversized CDC Transaction Count              | <p>Number of oversized transactions read by the CDC task. This metric applies only to Oracle tasks.</p> <p>It indicates the total number of oversized transactions (containing more than 100,000 data records) in the last period (5 minutes). For example, if a service should not have transactions with more than 100,000 data records, set the threshold to greater than or equal to 1.</p>                                                                                                                    |
|          | Timed-out CDC Transaction Count              | Number of timed-out transactions read by the CDC task.                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

## Metrics Supported by APIC

Table 12-2 Metrics supported by APIC

| Category | Metrics                      | Description                                         |
|----------|------------------------------|-----------------------------------------------------|
| Instance | Data API Calls               | Number of times that a data API has been called     |
|          | Maximum Latency for Data API | Maximum latency of a data API                       |
|          | Average Latency for Data API | Average latency of a data API                       |
|          | Data API Failures            | Number of times that a data API has been called     |
|          | Function API Calls           | Number of times that a function API has been called |

| Category | Metrics                          | Description                                         |
|----------|----------------------------------|-----------------------------------------------------|
|          | Maximum Latency for Function API | Maximum latency of a function API                   |
|          | Average Latency for Function API | Average latency of a function API                   |
|          | Function API Failures            | Number of times that a function API fails           |
|          | API Calls                        | Number of times that an API has been called         |
|          | 4xx Errors                       | Number of times that an API returns a 4xx error     |
|          | 5xx Errors                       | Number of times that an API returns a 5xx error     |
|          | Throttled API Calls              | Number of times that an API call has been throttled |
|          | Average Latency                  | Average latency of an API                           |
|          | Maximum Latency                  | Maximum latency of an API                           |
| API      | API Calls                        | Number of API calls                                 |
|          | 2xx Responses                    | Number of times that the API returns a 2xx response |
|          | 4xx Errors                       | Number of times that the API returns a 4xx error    |
|          | 5xx Errors                       | Number of times that the API returns a 5xx error    |
|          | Total Errors                     | Total number of API errors.                         |
|          | Average Latency                  | Average latency of the API                          |
|          | Maximum Latency                  | Maximum latency of the API                          |
|          | Incoming Traffic                 | Incoming traffic of the API                         |
|          | Outgoing Traffic                 | Outgoing traffic of the API                         |

## Metrics Supported by MQS

**Table 12-3** Metrics supported by MQS

| Category        | Metrics                                       | Description                                                                |
|-----------------|-----------------------------------------------|----------------------------------------------------------------------------|
| Instance        | Partitions                                    | Number of used partitions in an instance                                   |
|                 | Topics                                        | Number of created topics in an instance                                    |
|                 | Accumulated Messages                          | Total number of accumulated messages in all consumer groups of an instance |
| Broker          | Message Size                                  | Total size of messages in the broker                                       |
|                 | Message Creation Rate                         | Number of messages created per second                                      |
|                 | Message Retrieval                             | Number of bytes retrieved per second                                       |
|                 | Message Creation                              | Number of bytes created per second                                         |
|                 | Public Inbound Traffic                        | Inbound traffic over public networks per second of the broker              |
|                 | Public Outbound Traffic                       | Outbound traffic over public networks per second of the broker             |
|                 | Average Message Creation Processing Duration  | Average time that the broker spends processing message creation requests   |
|                 | Average Message Retrieval Processing Duration | Average time that the broker spends processing message retrieval requests  |
|                 | Broker Alive                                  | Whether the MQS broker is alive                                            |
|                 | Connections                                   | Total number of TCP connections on the MQS broker                          |
|                 | CPU Usage                                     | CPU usage of the MQS VM                                                    |
|                 | Average Disk Read Time                        | Average time for each disk I/O read in the monitoring period               |
|                 | Average Disk Write Time                       | Average time for each disk I/O write in the monitoring period              |
| Inbound Traffic | Inbound traffic per second of the MQS broker  |                                                                            |

| Category | Metrics                        | Description                                                                                  |
|----------|--------------------------------|----------------------------------------------------------------------------------------------|
|          | Outbound Traffic               | Outbound traffic per second of the MQS broker                                                |
|          | Average Load per CPU Core      | Average load of each CPU core of the MQS VM                                                  |
|          | Disk Capacity Usage            | Disk usage of the MQS VM                                                                     |
|          | Memory Usage                   | Memory usage of the MQS VM                                                                   |
|          | JVM Heap Memory Usage of Kafka | Heap memory usage of the MQS Kafka JVM                                                       |
|          | Successful Creation Requests   | Number of message creation requests successfully processed per minute by the Rest node       |
|          | Failed Creation Requests       | Number of message creation requests that failed to be processed per minute by the Rest node  |
|          | Creation Request Latency       | Average latency of message creation requests processed by the Rest node                      |
|          | Created Messages               | Number of messages created per minute by the Rest node                                       |
|          | Message Creation               | Total size of messages created per second by the Rest node                                   |
|          | Successful Retrieval Requests  | Number of message retrieval requests successfully processed per minute by the Rest node      |
|          | Failed Retrieval Requests      | Number of message retrieval requests that failed to be processed per minute by the Rest node |
|          | Retrieval Request Latency      | Average latency of message retrieval requests processed by the Rest node                     |
|          | Retrieved Messages             | Number of messages retrieved per minute by the Rest node                                     |
|          | Message Retrieval              | Total size of messages retrieved per second by the Rest node                                 |
|          | Successful Request Submissions | Number of requests successfully submitted per minute by the Rest node                        |

| Category       | Metrics                      | Description                                                                                       |
|----------------|------------------------------|---------------------------------------------------------------------------------------------------|
|                | Failed Request Submissions   | Number of requests that failed to be submitted per minute by the Rest node                        |
|                | Submission Request Latency   | Average latency of request submission by the Rest node                                            |
|                | Submitted Messages           | Number of messages submitted per minute by the Rest node                                          |
|                | Request Submission Rate      | Total size of requests submitted per second by the Rest node                                      |
| Queue          | Message Size                 | Total size of messages in the queue                                                               |
|                | Message Creation Rate        | Number of messages created per second                                                             |
|                | Message Retrieval            | Number of bytes retrieved per second                                                              |
|                | Message Creation             | Number of bytes created per second                                                                |
|                | Total Messages               | Total number of messages in the queue                                                             |
|                | Created Messages             | Number of messages that have been created                                                         |
|                | Partition Messages           | Total number of messages in the partition                                                         |
| Consumer group | Partition Retrieved Messages | Number of messages retrieved by the consumer group                                                |
|                | Partition Available Messages | Number of messages that can be retrieved in the consumer group                                    |
|                | Topic Available Messages     | Number of remaining messages that can be retrieved from the specified topic in the consumer group |
|                | Topic Retrieved Messages     | Number of messages that have been retrieved from the specified topic in the consumer group        |
|                | Consumer Available Messages  | Number of remaining messages that can be retrieved in the consumer group                          |

| Category | Metrics                     | Description                                                       |
|----------|-----------------------------|-------------------------------------------------------------------|
|          | Consumer Retrieved Messages | Number of messages that have been retrieved in the consumer group |

## Metrics Supported by LINK

**Table 12-4** Metrics Supported by LINK

| Metrics                             | Description                                                                                  |
|-------------------------------------|----------------------------------------------------------------------------------------------|
| Online Devices                      | Number of online devices of a user                                                           |
| Total Number of Messages            | Number of messages sent by all devices of a user                                             |
| TPS                                 | Number of messages sent by devices per second in a measurement period                        |
| Maximum Latency for Message Sending | Number of milliseconds for which a device delays sending of messages in a measurement period |

## Dimensions

| Key                     | Value                                               |
|-------------------------|-----------------------------------------------------|
| instance_id             | ROMA Connect instance                               |
| fdi                     | Data integration                                    |
| apic                    | Service integration                                 |
| kafka_instance_id       | Message integration instance                        |
| kafka_broker            | Message integration broker node                     |
| kafka_rest              | Message integration Rest node                       |
| kafka_topics            | Message integration queue                           |
| kafka_partitions        | Message integration partition                       |
| kafka_groups-partitions | Consumer group of the message integration partition |
| kafka_groups_topics     | Consumer group of the message integration queue     |

| Key          | Value                                 |
|--------------|---------------------------------------|
| kafka_groups | Consumer group of message integration |
| link         | Device integration                    |

# 13 Permissions

[Assigning ROMA Connect Permissions](#)

[ROMA Connect Custom Policies](#)

## 13.1 Assigning ROMA Connect Permissions

### Overview

This section describes how to use **IAM** to implement fine-grained permissions control for your ROMA Connect resources. With IAM, you can:

- Create IAM users for employees based on the organizational structure of your enterprise. Each IAM user has their own security credentials, providing access to ROMA Connect resources.
- Grant only the permissions required for users to perform a task.
- Entrust a Huawei Cloud account or cloud service to perform professional and efficient O&M on your ROMA Connect resources.

If your Huawei Cloud account does not need individual IAM users, then you may skip this chapter.

### Prerequisites

Learn about the permissions (see [Permissions Management](#)) supported by ROMA Connect and choose policies or roles according to your requirements. For the system-defined policies of other services, see [System Permissions](#).

### Creating a User and Assigning Permissions

1. [Create a user group and assign permissions](#) to it.  
Create a user group on the IAM console, and assign the **ROMA ReadOnlyAccess** policy to the group.
2. [Create an IAM user](#).  
Create a user on the IAM console and add the user to the group created in **1**.



3. **Log in** and verify permissions.  
Log in to the management console by using the user created, and verify that the user has the granted permissions.
  - Choose **Service List > Enterprise Application > Application & Data Integration Platform ROMA**. On the ROMA Connect console, click **Buy ROMA Instance** in the upper right corner. If a message appears indicating that you have insufficient permissions to perform the operation, the **ROMA ReadOnlyAccess** policy has already taken effect.
  - Choose any other service in **Service List**. If a message appears indicating that you have insufficient permissions to perform the operation, the **ROMA ReadOnlyAccess** policy has already taken effect.

## Assigning Permissions to an Existing User Group

1. **Assign permissions to a user group**.  
Select an existing user group and assign the **ROMA ReadOnlyAccess** policy to it.
2. **Log in** and verify permissions.  
Log in to the management console using a user in the user group in **1**, switch to the authorized region, and verify the user's permissions.  
Choose **Service List > Enterprise Application > Application & Data Integration Platform ROMA**. On the ROMA Connect console, click **Buy ROMA Instance** in the upper right corner. If a message appears indicating that you have insufficient permissions to perform the operation, the **ROMA ReadOnlyAccess** policy has already taken effect.

## 13.2 ROMA Connect Custom Policies

Create custom permissions policies for ROMA Connect if the preset ones cannot satisfy your needs. For the actions that can be added to custom policies, see [Permissions Policies and Supported Actions](#).

You can create custom policies in either of the following ways:

- Visual editor: Select cloud services, actions, resources, and request conditions. This does not require knowledge of policy syntax.
- JSON: Edit JSON policies from scratch or based on an existing policy.

For details about how to create a custom policy, see [Creating a Custom Policy](#). This section provides examples of common custom policies of ROMA Connect.

### Example Custom Policies

- Example 1: Allow users to create, start/stop, and view integration tasks.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "roma:tasks:create",
        "roma:tasks:operate",
        "roma:tasks:list",
        "roma:tasks:get"
      ]
    }
  ]
}
```

```
    ]  
  }  
]  
}
```

- Example 2: Grant users all permissions for service integration, including connector permissions.

```
{  
  "Version": "1.1",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "roma:tasks:*",  
        "roma:connectors:*"  
      ]  
    }  
  ]  
}
```

- Example 3: Deny the deletion of ROMA Connect instances.

A policy with only "Deny" permissions must be used in conjunction with other policies to take effect. If the permissions assigned to a user contain both "Allow" and "Deny", the "Deny" permissions take precedence over the "Allow" permissions.

The following method can be used if you need to assign permissions of the **ROMA FullAccess** policy to a user but you want to prevent the user from deleting instances. Create a custom policy for denying instance deletion, and attach both policies to the group to which the user belongs. Then, the user can perform all operations on ROMA Connect except deleting instances. The following is an example of a deny policy:

```
{  
  "Version": "1.1",  
  "Statement": [  
    {  
      "Effect": "Deny",  
      "Action": [  
        "roma:instances:delete"  
      ]  
    }  
  ]  
}
```

- Example 4: Allow users to view and perform operations only on data integration tasks created by themselves.

To do so, you can add conditions to the custom policy so that the authorized items take effect only for the resources created by yourself. The following is an example of a condition policy:

```
{  
  "Version": "1.1",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "roma:tasks:create",  
        "roma:tasks:operate",  
      ]  
    }  
    {  
      "Effect": "Allow",  
      "Action": [  
        "roma:tasks:get",  
        "roma:tasks:list"  
      ],  
    }  
  ],  
}
```

```
    "Condition": {
      "StringEquals": {
        "roma:ResourceCreator": [
          "${g:UserId}"
        ]
      }
    }
  ]
}
```

## Example Custom Policy for ROMA Business Flows

Example 1: Authorize a user to create, query, and update business flow graphs but not to manage or delete them.

```
{
  "Version": "1.1",
  "Statement": [
    {
      "Action": [
        "romabfs:graph:get",
        "roma:graph:list",
        "roma:graph:update"
      ],
      "Effect": "Allow"
    }
  ]
}
```