

腾讯轻联 控制台指南（旧版）



腾讯云

【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【 商标声明 】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。

您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【 联系我们 】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

控制台指南（旧版）

主页

集成开发

集成应用

应用管理

应用配置

逻辑组件使用指南

Async

Break

Cache

Choice

Continue

Crypto

Flow Reference

For Each

Filter

Logger

Mapper

Parallel Foreach

Raise Error

Remove Variable

RecordSet Encoder

Scatter Gather

Scheduler

Set Payload

Set Variable

Synchronized

Transform

Try

Table

Until Successful

While

Compress

Rate Limiter

API 中心

API 管理

API 用户中心

运维中心

监控管理

运行日志

告警配置

告警策略

通知模板

通知模板

接口回调

告警接收渠道

告警历史

集成工具

连接管理

安全网关

通用存储

连接器

连接器市场

连接器开发

连接器使用指南

Apache Kafka 连接器

Apache Kafka 连接器使用指南

阿里云 Kafka 接入腾讯云数据连接器

华为云 Kafka 接入腾讯云数据连接器

腾讯云 CKafka 接入腾讯云数据连接器

Database 连接器

Database 连接器使用指南

华为云 MySQL 接入腾讯云数据连接器

腾讯云 MySQL 接入腾讯云数据连接器

Redis 连接器

Redis 连接器使用指南

阿里云 Redis 接入腾讯云数据连接器

华为云 Redis 接入腾讯云数据连接器

腾讯云 Redis 接入腾讯云数据连接器

HTTP Listener 连接器

HTTP Request 连接器

SOAP 连接器

AMQP 连接器

MongoDB 连接器

FTP 连接器

SFTP 连接器

Email 连接器

状态存储连接器

Neo4j 连接器

JWT 连接器

GRPC 连接器

MQTT 连接器

MySQL Change Data Capture 触发器

管理中心

成员管理

项目管理

环境管理

审计日志

DataWay 表达式

简介

快速入门

函数参考

开发指南

DataWay 语言基础概念

使用 Entity 对象

MimeType 支持

DataWay 脚本调试

场景案例

错误码说明

常见问题

Cron 表达式介绍

访问管理示例

控制台指南（旧版）

主页

最近更新时间：2024-09-10 16:00:11

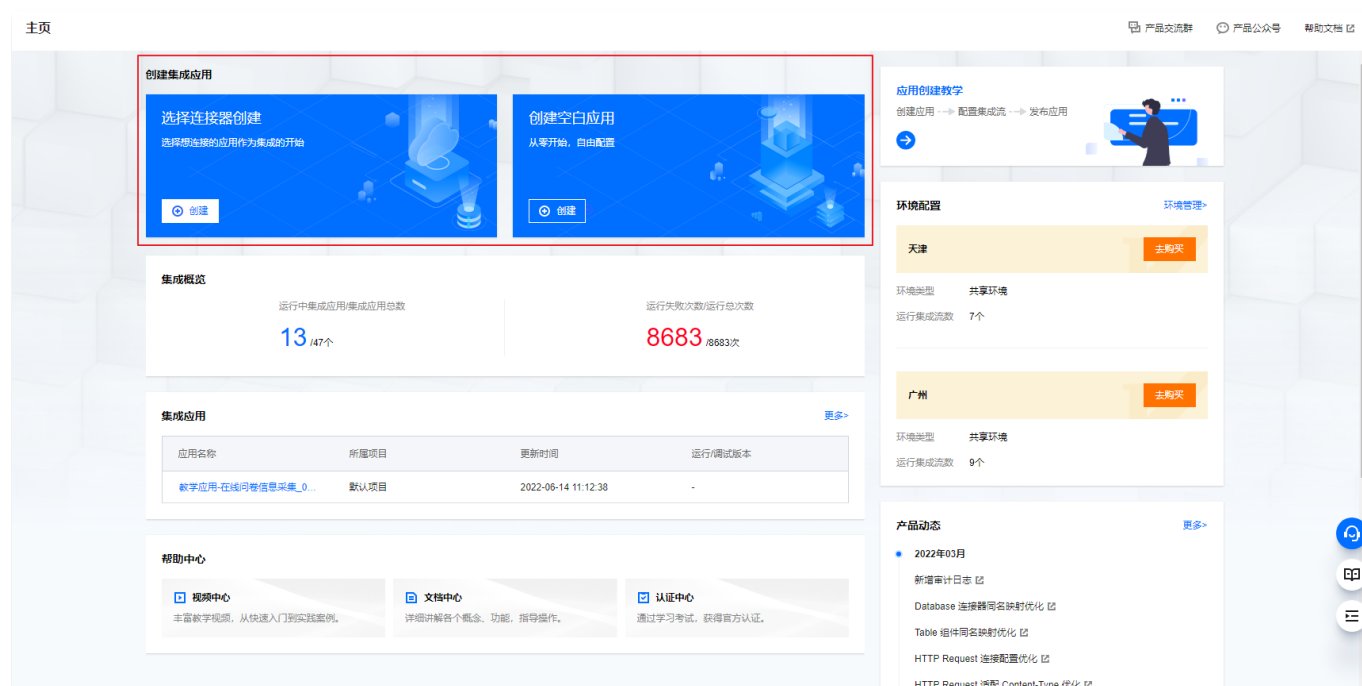
腾讯云数据连接器 [主页](#) 集中展示了创建集成应用、集成概览、帮助中心及应用创建教学等内容，本文将展开详细介绍。

注意：

新用户访问主页链接会直接跳转到新版控制台，您可前往新版控制台操作使用，详情请参见 [新版控制台](#)。

创建集成应用

主页提供了快速创建集成应用的入口，分为两种方式：选择连接器创建和创建空白应用。



选择连接器创建

在选择连接器创建中，您可以选择想连接的应用作为集成的开始来创建集成应用。

1. 在创建集成应用的选择连接器创建模块，单击创建。

主页

创建集成应用

选择连接器创建

选择想连接的应用作为集成的开始

创建

创建空白应用

从零开始，自由配置

创建

2. 填写应用名称，并选择想连接的应用，单击下一步。

通过连接器创建

应用名称 *

请输入25位以内的中文、字母、数字、_或-

选择一个开始的应用

请输入关键词搜索

腾讯云...

腾讯微卡

壹云汇

QQ邮箱

至信链

convertl...

腾讯问卷

通途ER...

腾讯乐享

维格表

腾讯企点

腾讯广告

普客

六度人...

拓课云

Phenom

下一步

取消

3. 选择连接器对应的触发或执行操作（此处以企业微信为例，选择不同的应用会展示不同的操作），单击下一步。

企业微信

通讯录管理/成员

消息推送

消息推送/群聊全

通讯录管理/部门

客户联系/企业版

客户联系/客户管

客户联系/客户标

客户联系/在职继

客户联系/离职继

客户联系/客户群

客户联系/客户朋

客户联系/消息推

客户联系/统计推

搜索

创建成员

读取成员

更新成员

删除成员

批量删除成员

获取部门成员

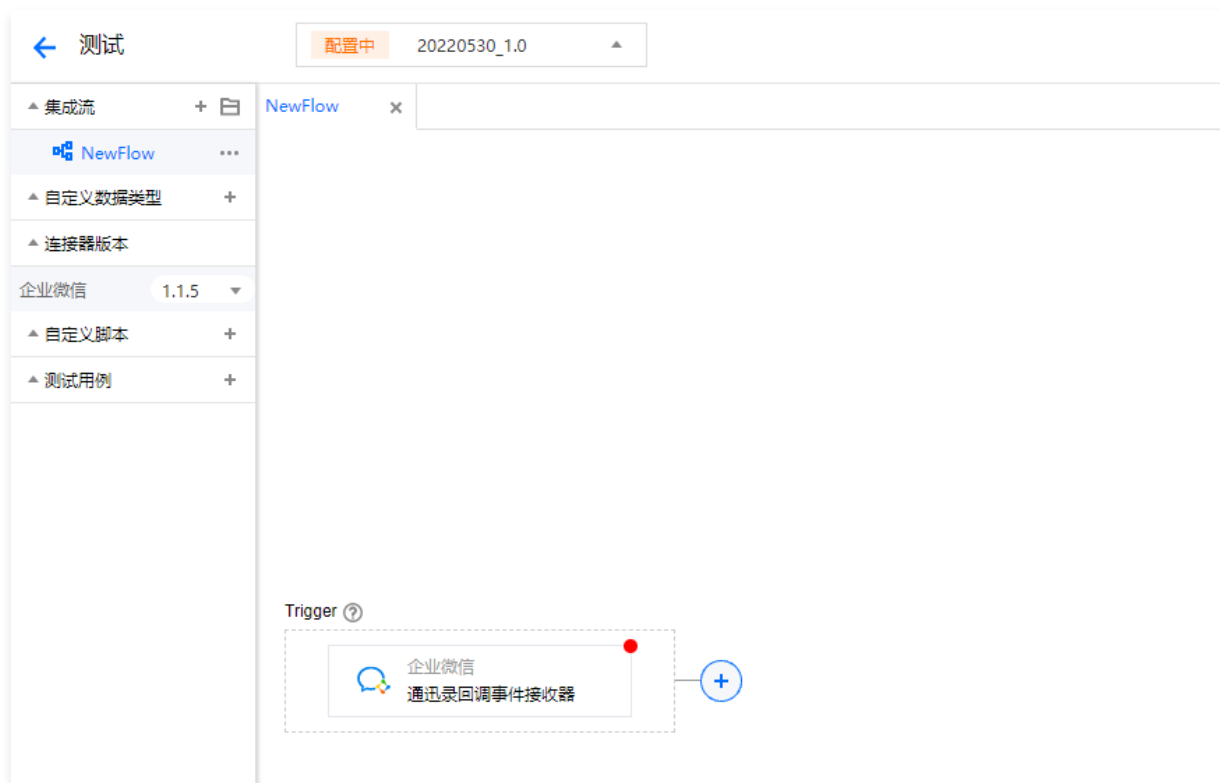
获取部门成员详情

手机号获取用户ID

用户ID转OpenId

OpenId转用户ID

4. 进入集成流开发界面继续配置集成流，可参见 [应用配置](#)。



创建空白应用

在创建空白应用中，您可以从零开始，自由配置。

1. 在创建集成应用的创建空白应用模块，单击创建。

主页

创建集成应用

选择连接器创建

选择想连接的应用作为集成的开始

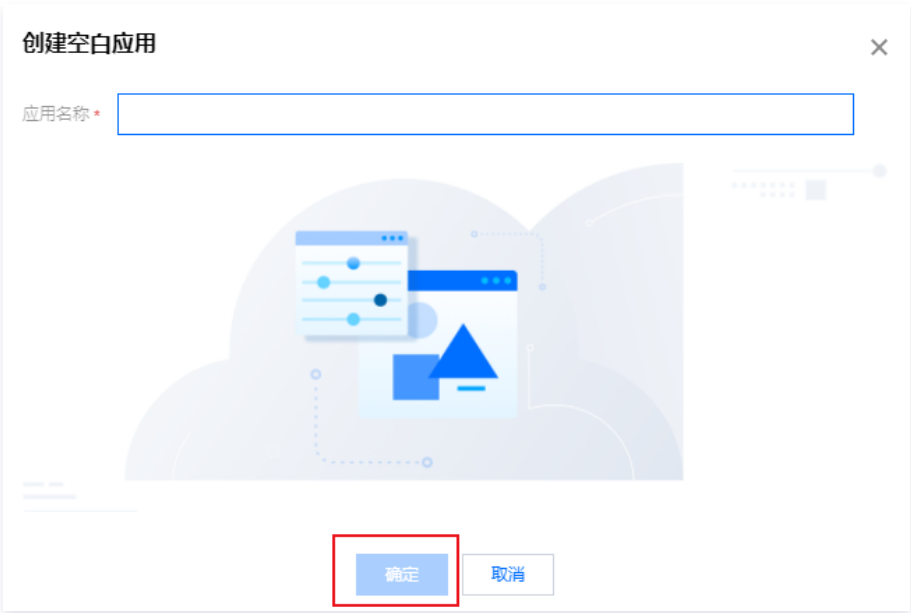
创建

创建空白应用

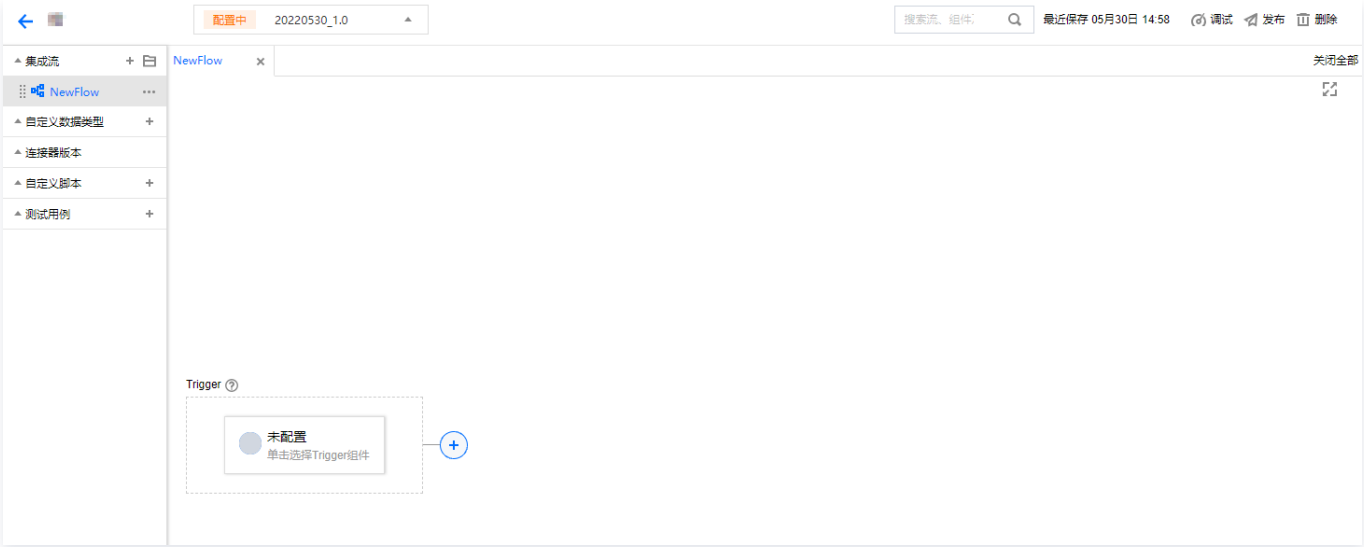
从零开始，自由配置

创建

2. 填写应用名称，单击完成。



3. 进入集成流开发界面配置集成流，可参见 [应用配置](#)。



集成概览

主页中的集成概览汇总展示了对应账号下，集成应用总数、处于运行中的集成应用数量、运行总次数以及运行失败总次数方便用户直观的了解集成应用的基本运行情

主页

创建集成应用

选择连接器创建

选择想连接的应用作为集成的开始

创建

创建空白应用

从零开始，自由配置

创建

集成概览

运行中集成应用/集成应用总数

13 /47个

运行失败次数/运行总次数

8683 /8683次

集成应用

更多>

应用名称	所属项目	更新时间	运行/调试版本
教学应用-在线问卷信息采集_0...	默认项目	2022-06-14 11:12:38	-

帮助中心

视频中心

丰富教学视频，从快速入门到实践案例。

文档中心

详细讲解各个概念、功能、指导操作。

认证中心

通过学习考试，获得官方认证。

应用创建教学

创建应用 --> 配置集成流 --> 发布应用

环境配置

环境管理>

天津

去购买

环境类型 共享环境

运行集成流数 7个

广州

去购买

环境类型 共享环境

运行集成流数 9个

产品动态

更多>

2022年03月

新增审计日志

Database 连接器同名映射优化

Table 组件同名映射优化

HTTP Request 连接器配置优化

HTTP Request 适配 Content-Type 优化

况。

集成应用

主页中的集成应用展示了对应账户下部分应用的基本信息，按照更新时间展示最近更新五个应用的信息，若想查询账号下所有应用信息，单击更多即可。

主页

创建集成应用

选择连接器创建

选择想连接的应用作为集成的开始

创建

创建空白应用

从零开始，自由配置

创建

集成概览

运行中集成应用/集成应用总数

13 /47个

运行失败次数/运行总次数

8683 /8683次

集成应用

更多>

应用名称	所属项目	更新时间	运行/调试版本
教学应用-在线问卷信息采集_0...	默认项目	2022-06-14 11:12:38	-

帮助中心

视频中心

丰富教学视频，从快速入门到实践案例。

文档中心

详细讲解各个概念、功能、指导操作。

认证中心

通过学习考试，获得官方认证。

应用创建教学

创建应用 --> 配置集成流 --> 发布应用

环境配置

环境管理>

天津

去购买

环境类型 共享环境

运行集成流数 7个

广州

去购买

环境类型 共享环境

运行集成流数 9个

产品动态

更多>

2022年03月

新增审计日志

Database 连接器同名映射优化

Table 组件同名映射优化

HTTP Request 连接器配置优化

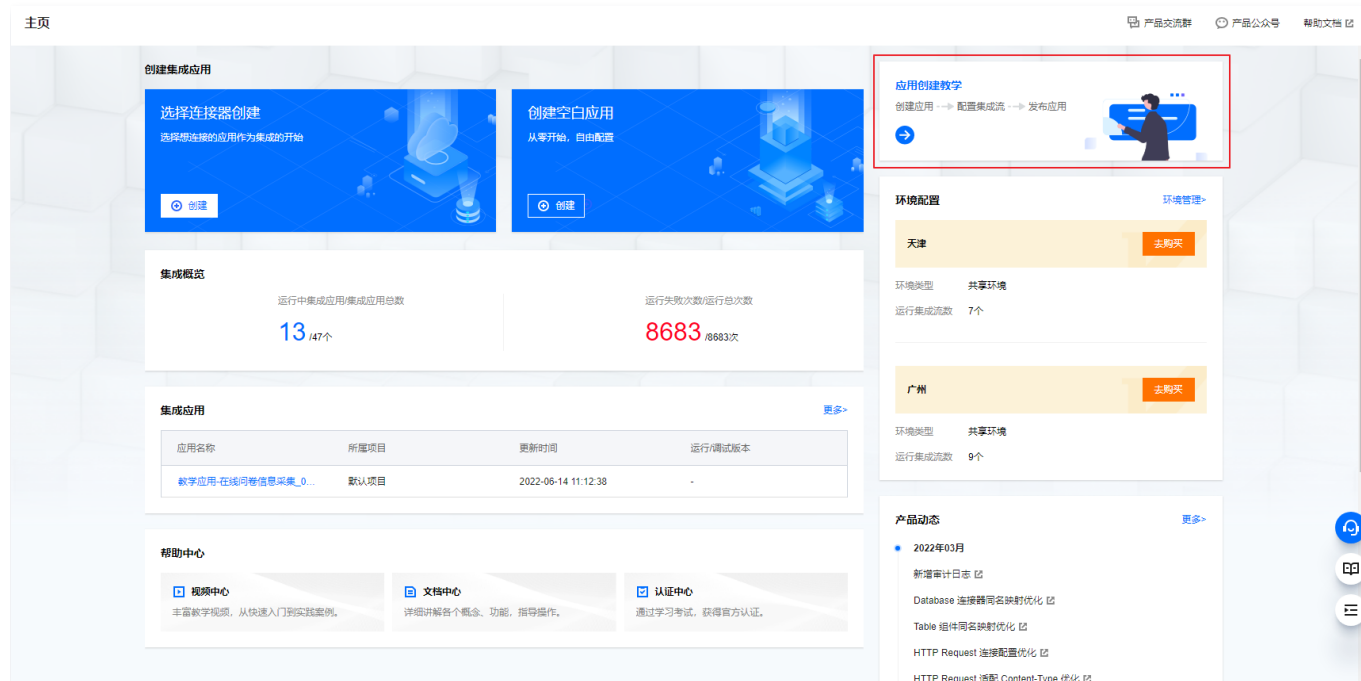
HTTP Request 适配 Content-Type 优化

应用创建教学

版权所有：腾讯云计算（北京）有限责任公司

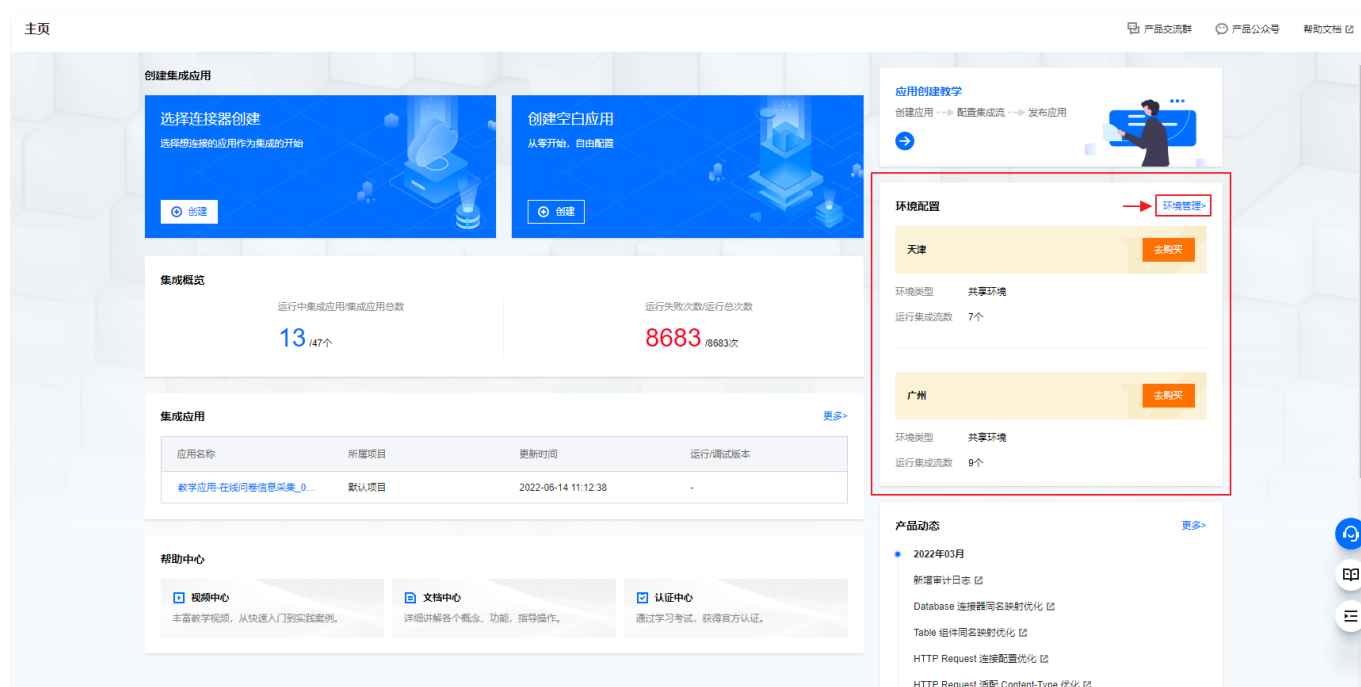
第9 共732页

提供动态的应用创建教学案例指引，协助新用户尽快熟悉了解腾讯云数据连接器的基本概念及操作。也可参见 [应用创建教学指南](#)。



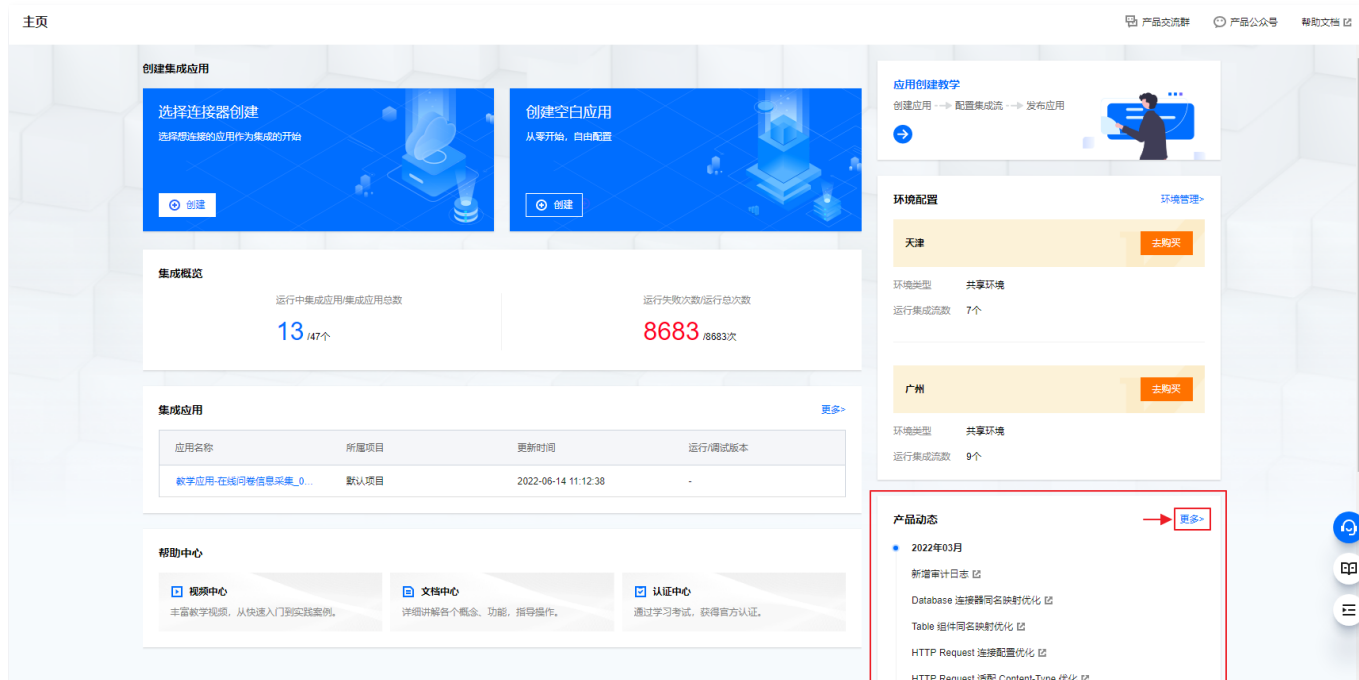
环境配置

主页中的环境配置展示了对应账户下部分环境的基本信息，包含环境名称、环境类型、环境中运行的集成流数量，若想查询账号下所有的环境信息，单击环境管理即可，更多环境管理相关内容可参见 [环境管理](#)。



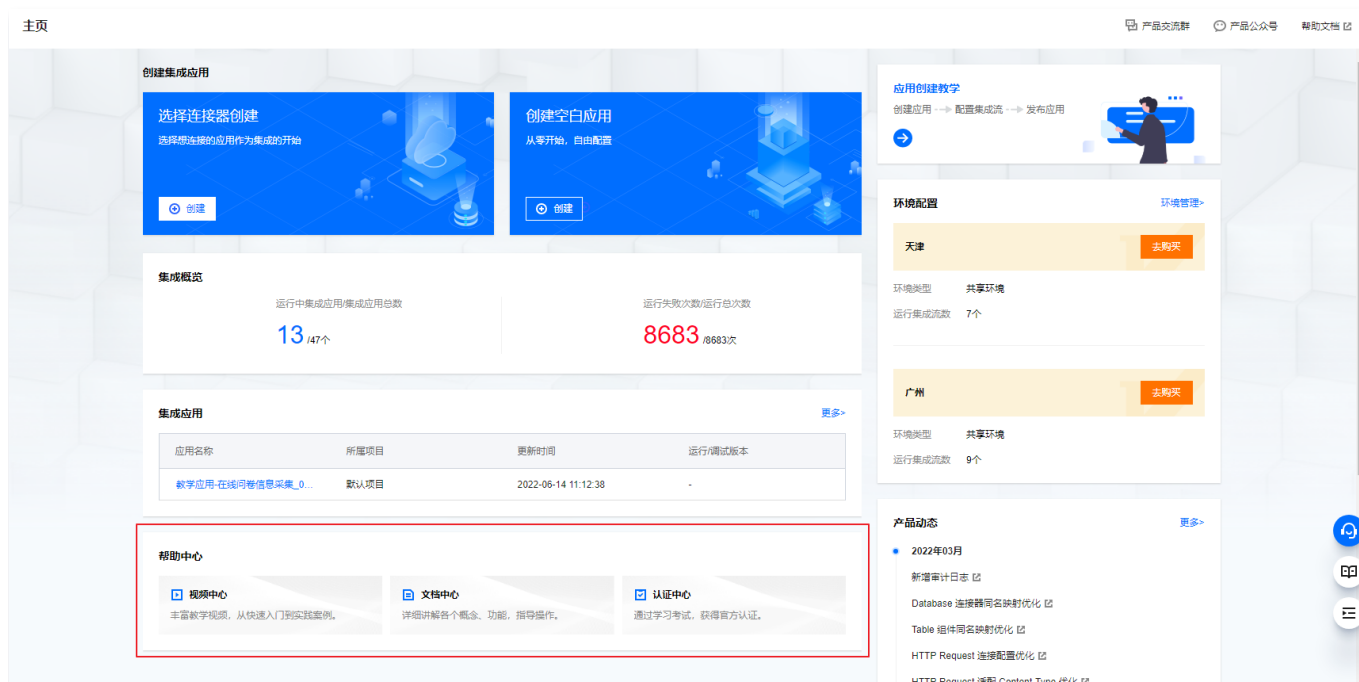
产品动态

主页中的产品动态展示了产品最近新增或更新的文档或功能等相关信息，方便用户更直观的了解产品的新增特性。若需了解更多历史更新内容，可单击更多进入 [产品动态](#) 进行查看。



帮助中心

主页中的帮助中心，为用户提供了进一步了解腾讯云数据连接器的学习视频，使用文档。对数据连接器使用及概念比较熟悉的用户也可以在通过认证中心的考试后获得数据连接器的官方认证。



更多操作

在主页右侧导航上，您可以通过以下方式了解更多产品信息：

- 单击产品交流群，扫码加入数据连接器交流群，您在使用产品时遇到任何问题都可以在群内反馈，会有技术专员跟进解答。
- 单击产品公众号，扫码关注腾讯云数据连接器公众号，获取最新产品动态。
- 单击帮助文档，查看数据连接器的所有产品文档。

主页

产品交流群 产品公众号 帮助文档

创建集成应用

选择连接器创建

选择您连接的应用作为集成的开始

创建

创建空白应用

从零开始，自由配置

创建

集成概览

运行中集成应用/集成应用总数

13 / 47个

运行失败次数/运行总次数

8683 / 8683次

集成应用

更多>

应用名称	所属项目	更新时间	运行/调试版本
教学应用-在线问答信息采集_0...	默认项目	2022-06-14 11:12:38	-

帮助中心

视频中心

丰富教学视频，从快速入门到实践案例。

文档中心

详细讲解各个概念、功能、指导操作。

认证中心

通过学习考试，获得官方认证。

应用创建教学

创建应用 → 配置集成流 → 发布应用

→

环境配置

环境管理>

天津

去购买

环境类型 共享环境

运行集成流数 7个

广州

去购买

环境类型 共享环境

运行集成流数 9个

产品动态

更多>

2022年03月

新增审计日志

Database 连接器同名映射优化

Table 组件同名映射优化

HTTP Request 连接配置优化

HTTP Request 适配 Content-Type 优化

集成开发

集成应用

应用管理

最近更新时间：2024-08-22 14:27:51

操作场景

集成需求可以通过创建应用集成进行实施，复杂的场景可创建多个应用集成，每个应用集成对应一个集成场景。进入应用集成页面，您可以操作添加应用、配置应用、修改应用名称、导出导入应用以及删除应用等操作。

不同的连接器状态，对应“操作”下展示的功能是有区别的，具体如下：

状态	功能
运行中	允许用户进行查看、修改名称、复制、导出和停止。
配置中	允许用户进行查看、修改名称、导出、发布和删除。
已停止	允许用户查看、配置、修改名称、发布和删除。
调试中	允许用户查看、配置、修改名称、导出、应用测试、单元测试和退出调试模式。

操作步骤

添加应用

当需要新建业务流时，您可按以下步骤操作：

- 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**应用集成**。
- 在**应用集成 > 应用管理**页面，选择对应项目名称后单击**添加应用**。



- 在添加应用页面，填选应用名称和添加方式后，单击**确定**，进入配置应用页面。

配置应用

当需要查看应用详情或修改配置时，您可按以下步骤操作：

- 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**应用集成**。

2. 在应用集成 > 应用管理页面，找到目标应用，单击配置或单击“应用名称”即可进入应用详情页面进行配置。

应用集成

默认项目

搜索应用、流、组件及配置

帮助文档

应用管理

添加应用

导入应用

应用名称	创建时间	更新时间	运行/调试版本	操作
test	2021-12-20 14:42:06	2021-12-20 14:42:07	-	配置 修改名称 删除 导出
HelloWorld	2021-12-10 11:28:21	2021-12-10 11:29:48	-	配置 修改名称 删除 导出

共 2 条

10 条 / 页

1 / 1 页

修改名称

当应用有变更，需要修改名称方便区分时，您可按以下步骤操作：

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击应用集成。
2. 在应用集成 > 应用管理页面，找到目标应用，单击修改名称，按照提示输入修改后的名称即可。

应用集成

默认项目

搜索应用、流、组件及配置

帮助文档

应用管理

添加应用

导入应用

应用名称	创建时间	更新时间	运行/调试版本	操作
test	2021-12-20 14:42:06	2021-12-20 14:42:07	-	配置 修改名称 删除 导出
HelloWorld	2021-12-10 11:28:21	2021-12-10 11:29:48	-	配置 修改名称 删除 导出

共 2 条

10 条 / 页

1 / 1 页

导入和导出应用

当您想和其他账号共享应用或者修改应用所属项目时（目前暂不支持直接修改应用所属项目及跨账号共享等操作），您可以使用导出导入功能。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击应用集成。
2. 在应用集成 > 应用管理页面，找到目标应用，单击左上角导入应用或应用项目操作列的导出即可完成操作。
 - 导出时可以选择不同版本的全部导出或选择需要的集成流和连接器配置导出，导出后以 .ipaas 为后缀的文件将自动下载。
 - 导入时只需要上传以 .ipaas 为后缀的文件即可。

应用集成

默认项目

搜索应用、流、组件及配置

帮助文档

应用管理

添加应用

导入应用

应用名称	创建时间	更新时间	运行/调试版本	操作
test	2021-12-20 14:42:06	2021-12-20 14:42:07	-	配置 修改名称 删除 导出
HelloWorld	2021-12-10 11:28:21	2021-12-10 11:29:48	-	配置 修改名称 删除 导出

共 2 条

10 条 / 页

1 / 1 页

删除应用

当应用处于已停止，且后续无需使用时，您可以删除应用。

注意

删除后数据无保留，请删除前导出做好备份或确认数据必要性再进行操作。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击应用集成。
2. 在应用集成 > 应用管理页面，找到目标应用，单击删除即可删除应用。

应用集成

默认项目

搜索应用、流、组件及配置

帮助文档

应用管理

添加应用

导入应用

应用名称	创建时间	更新时间	运行/调试版本	操作
test	2021-12-20 14:42:06	2021-12-20 14:42:07	-	配置 修改名称 删除 导出
HelloWorld	2021-12-10 11:28:21	2021-12-10 11:29:48	-	配置 修改名称 删除 导出

共 2 条

10 条 / 页

1 / 1 页

运行日志

当应用发布于运行中时，您可以通过以下步骤查看运行日志：

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击应用集成。
2. 在应用集成 > 应用管理页面，找到目标应用，单击操作列的运行日志。

应用集成

默认项目

搜索应用、流、组件及配置

帮助文档

应用管理

添加应用

导入应用

应用名称	创建时间	更新时间	运行/调试版本	操作
logDemo	2021-12-15 10:46:50	2021-12-15 10:50:14	运行中: 20211215_1.0(天津) ...	配置 修改名称 删除 导出 运行日志
组织同步	2021-08-20 15:14:14	2021-09-01 21:12:21	-	配置 修改名称 删除 导出
体验	2021-08-19 16:54:01	2021-08-20 12:20:54	-	配置 修改名称 删除 导出
World	2021-08-12 20:51:56	2021-08-14 13:48:10	-	配置 修改名称 删除 导出
	2021-08-12 14:39:19	2021-12-14 19:59:51	-	配置 修改名称 删除 导出

共 5 条

10 条 / 页

1 / 1 页

3. 在运行日志页面，您可查看当前运行应用、当前运行版本的近30分钟的日志。

运行监控

默认项目

[帮助文档](#)

监控

运行日志

定时任务管理

错误统计

天津

深度集成

logDemo

20211215_1.0

全部流

请选择日志级别

请输入作业标识

请输入日志内容关键词

近30分钟

搜索

日志数量 8776条

下载日志

时间	日志级别	应用名称	作业标识	集成流	组件/连接器	操作	输入/输出	日志内容
2021-12-15 17:44:12.468	INFO	logDemo	ffd2efdc-26d8-a95e-dc46-44d7abdd135a	NewFlow	timer		app:project.startFlow	triggerBy:
2021-12-15 17:44:12.469	INFO	logDemo	ffd2efdc-26d8-a95e-dc46-44d7abdd135a	NewFlow	timer		app:project.startFlow	triggerBy:
2021-12-15 17:44:12.474	WARN	logDemo	ffd2efdc-26d8-a95e-dc46-44d7abdd135a	NewFlow	logger		警告	可能一会就出问题！！
2021-12-15 17:44:12.474	ERROR	logDemo	ffd2efdc-26d8-a95e-dc46-44d7abdd135a	NewFlow	logger		出错了	出错了！！
2021-12-15 17:44:12.475	INFO	logDemo	ffd2efdc-26d8-a95e-dc46-44d7abdd135a	NewFlow	timer		app:connector.inbound	componentN
2021-12-15 17:44:13.656	INFO	logDemo	462dac26-5cca-c29c-3fe7-8010938ac6d3	NewFlow	timer		app:project.startFlow	triggerBy:
2021-12-15 17:44:13.658	INFO	logDemo	462dac26-5cca-c29c-3fe7-8010938ac6d3	NewFlow	timer		app:project.startFlow	triggerBy:

应用配置

最近更新时间：2024-08-22 14:34:21

以下视频将为您介绍集成流的入门、调试发布的相关说明：

集成流入门

[观看视频](#)

集成流调试发布

[观看视频](#)

操作场景

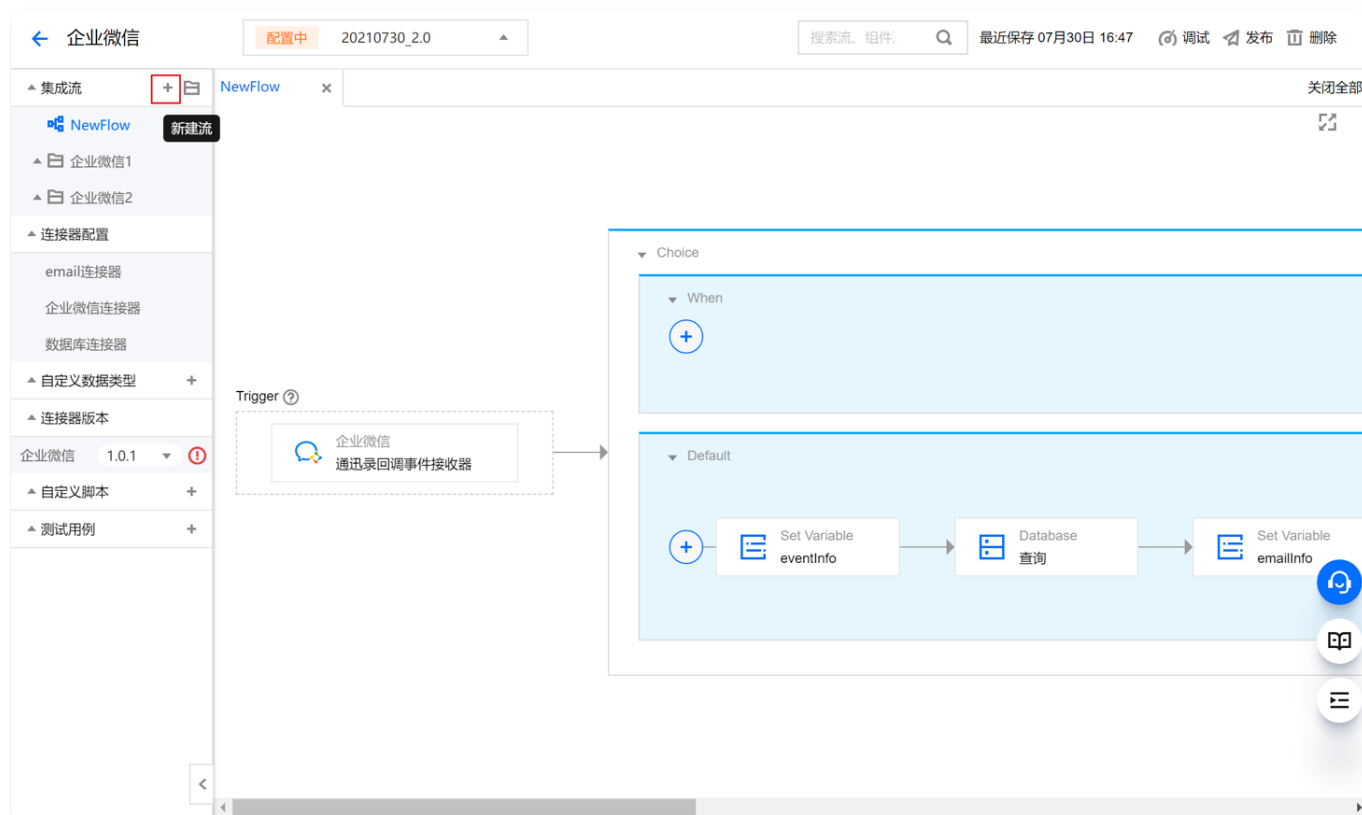
进入一个集成应用后，可以通过创建多条集成流来实现业务逻辑。集成流画布中预留触发类组件位，可以放置拥有触发能力的逻辑组件或连接器，例如：HTTP Listener、Kafka、定时器等，拥有触发能力的集成流为主流，不具备触发能力的集成流则只能被其他集成流通过逻辑组件 [Flow Reference](#) 引用为子流。

集成流

平台支持对集成流的层次化管理，可通过创建文件夹，将相关集成流聚合，且文件夹和集成流在同一应用内支持新建、复制等操作进一步方便开发及维护。

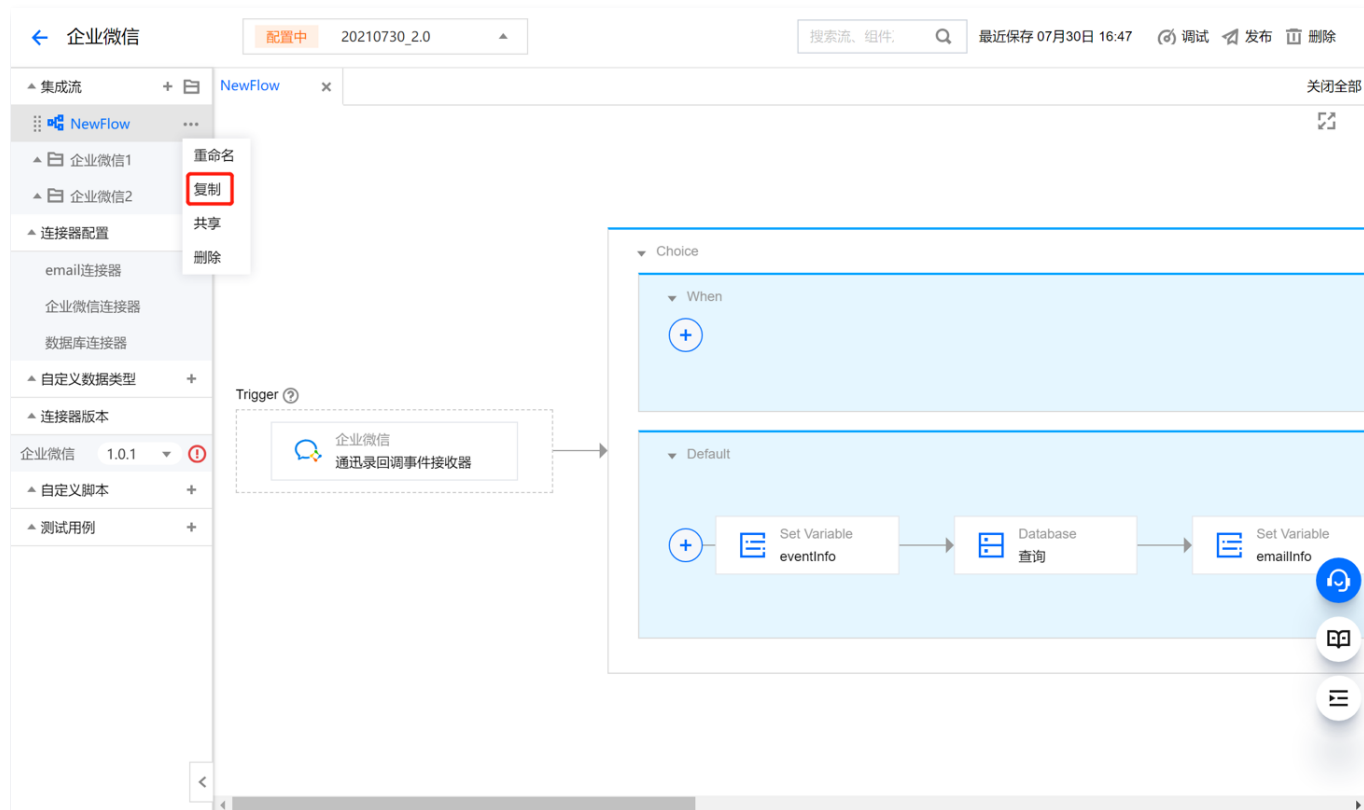
新建集成流

集成流支持新建、复制、共享、重命名及删除操作。



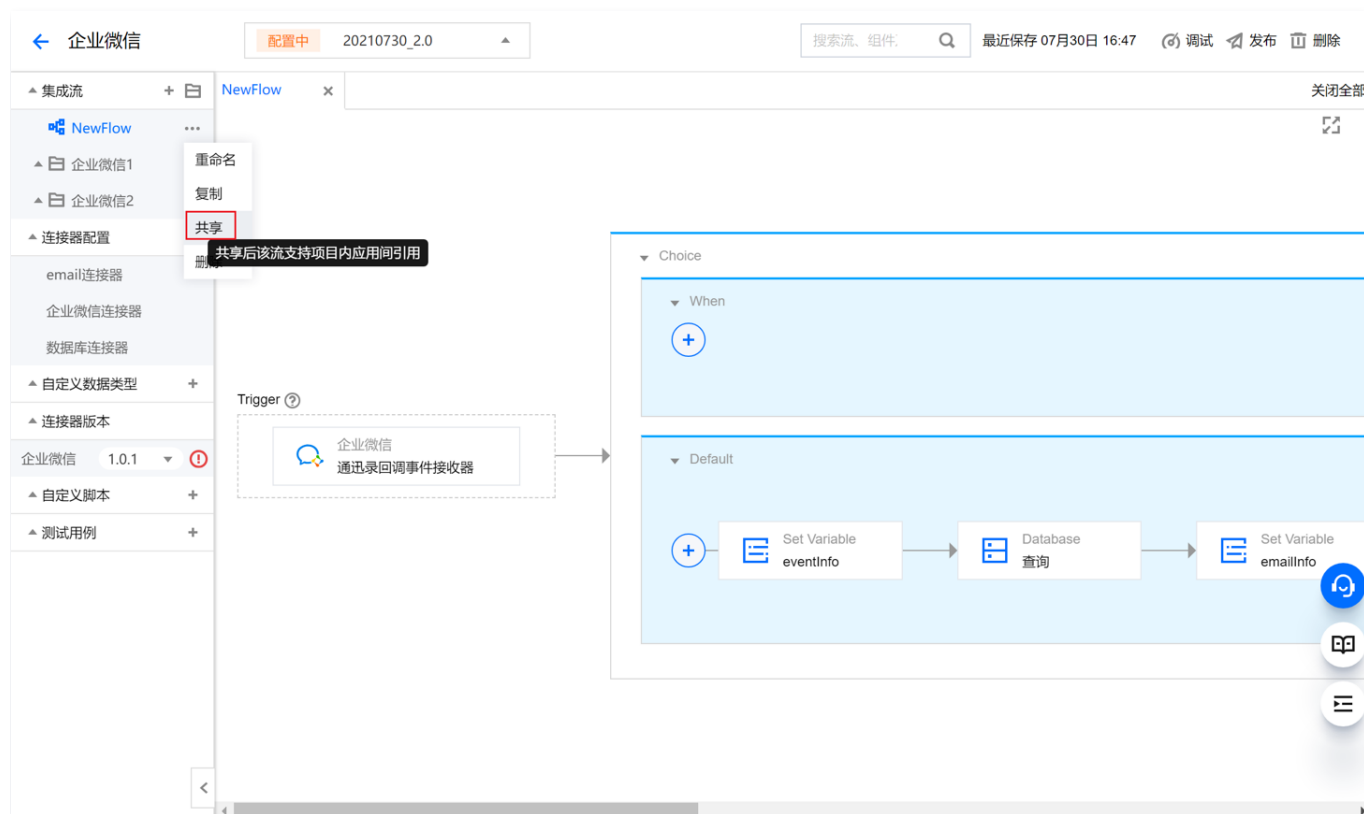
复制集成流

集成流可在同一应用内进行复制和粘贴操作。



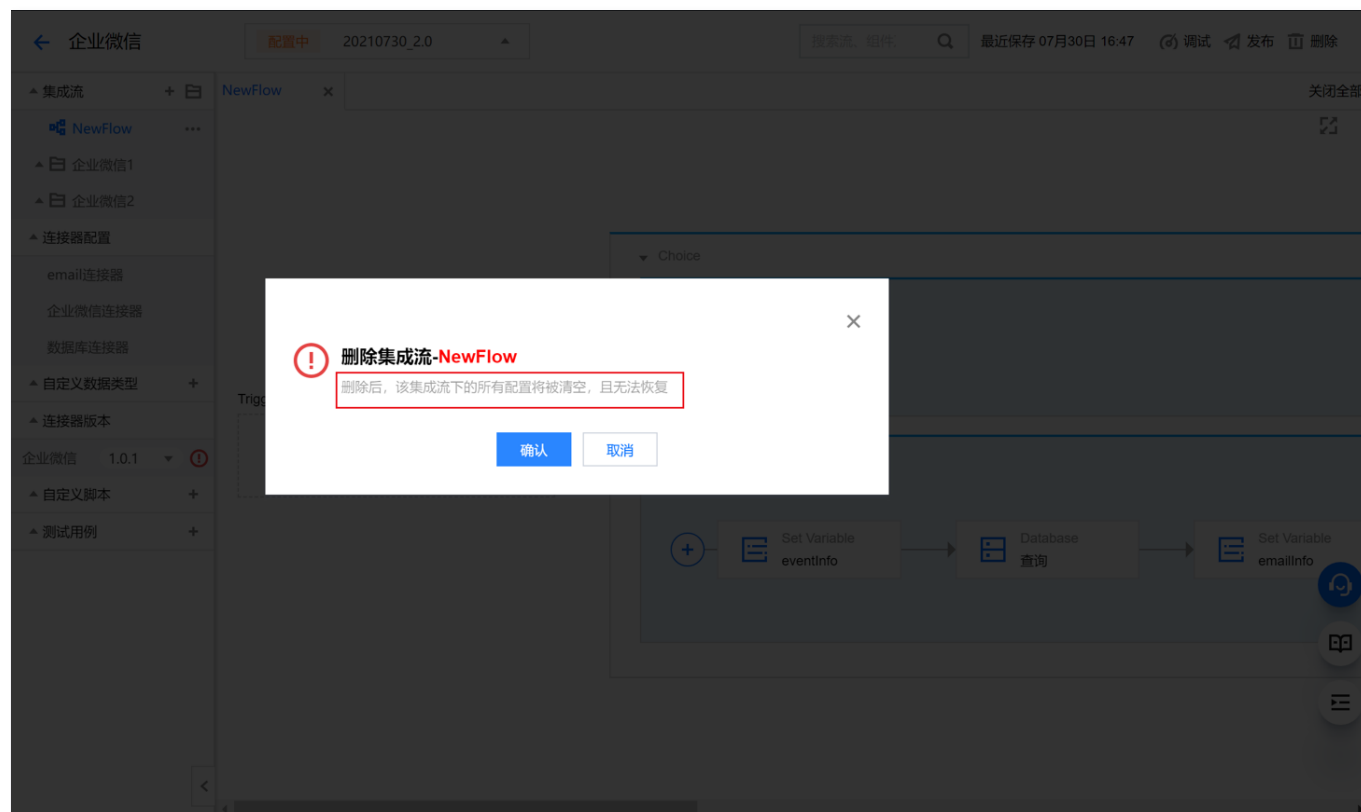
共享集成流

共享集成流后，该集成流支持项目内应用间引用。



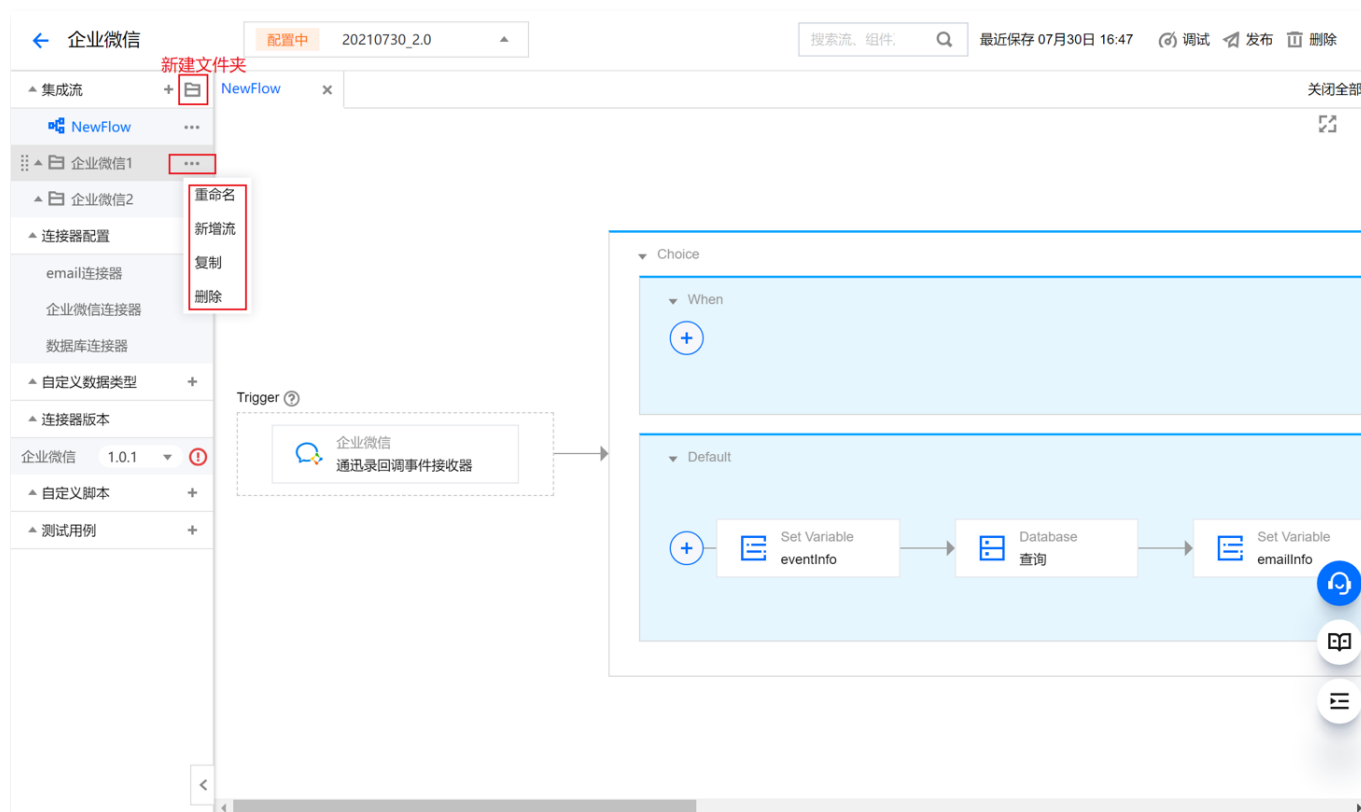
删除集成流

删除后，该集成流下的所有配置将被清空，且无法恢复，请谨慎操作。



文件夹

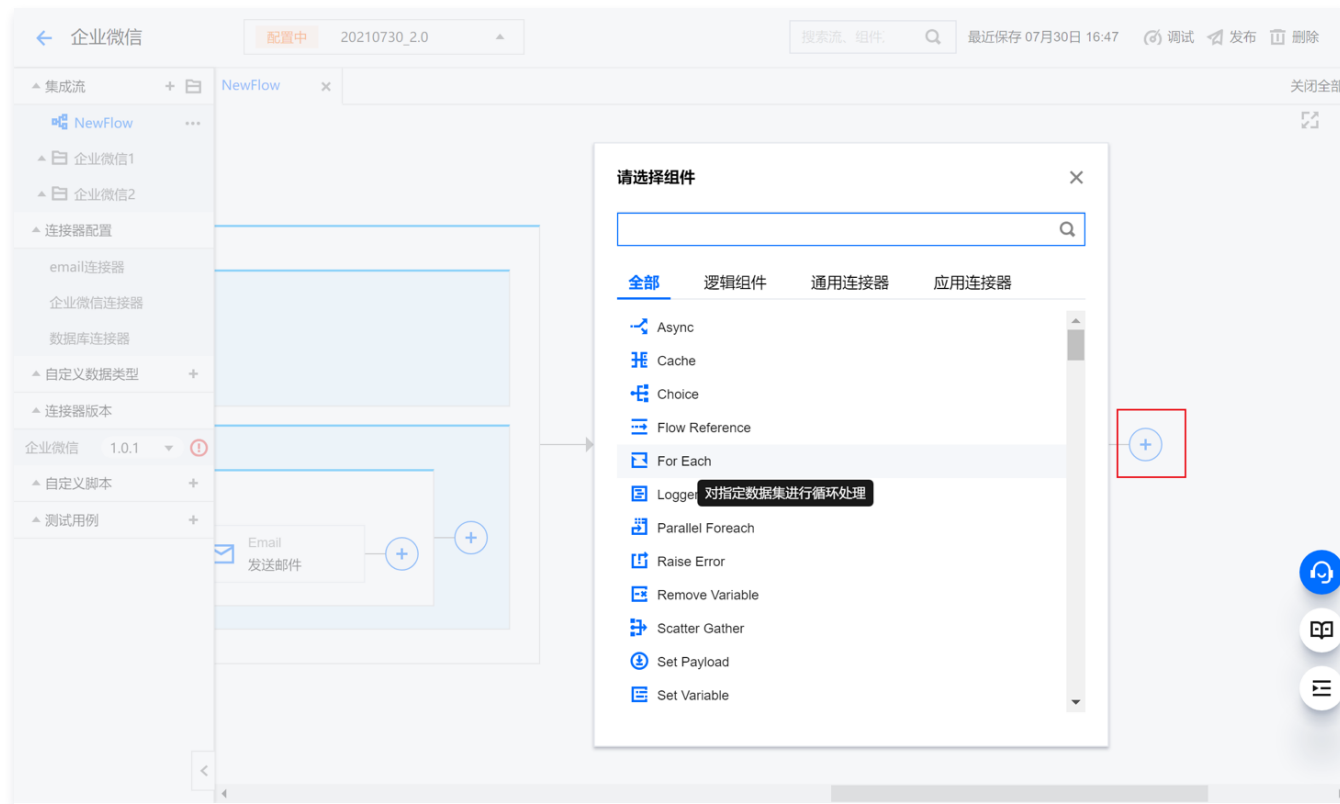
文件夹可将相关集成流聚合，且文件夹支持新建、新增流、复制等操作，方便您进一步管理维护集成流。



组件库

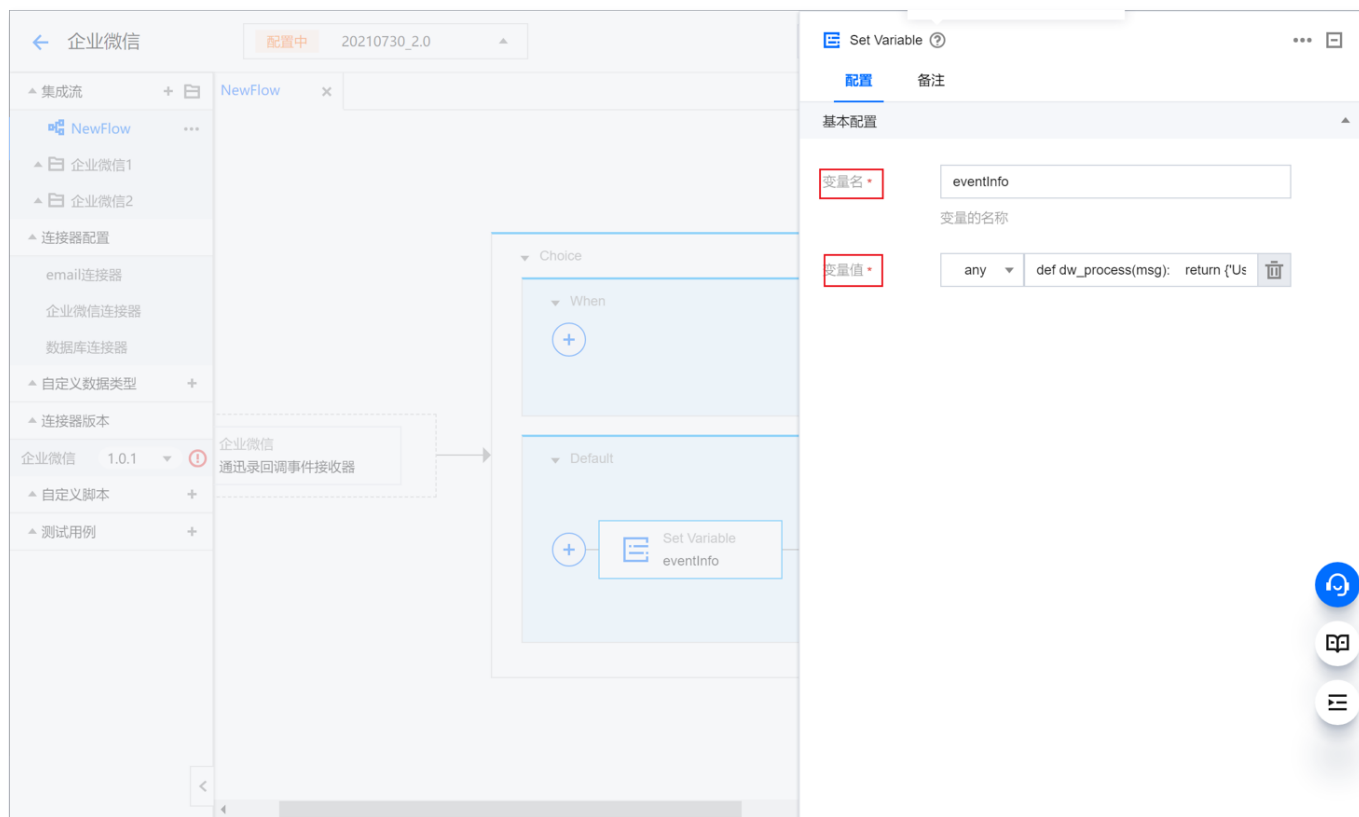
新建一个集成流后会在画布中展示，单击画布中的 + 可以打开组件库，选择需要组件配置集成流。组件包括连接器和逻辑组件。其中，连接器又包括通用连接器、应用连接器和逻辑组件。

- 通用连接器：对应用系统交互的协议。
- 应用连接器：对应用本身的封装。例如：一个系统通过 HTTP 与外界交互，可以选择 HTTP 连接器进行参数配置实现连接；若想连接腾讯会议，则可以直接使用腾讯会议的连接器（封装了腾讯会议的接口、鉴权和交互协议）。
- 逻辑组件用来实现两个应用对接时的业务逻辑。例如：循环、选择、设置变量、数据转换、并行异步等。



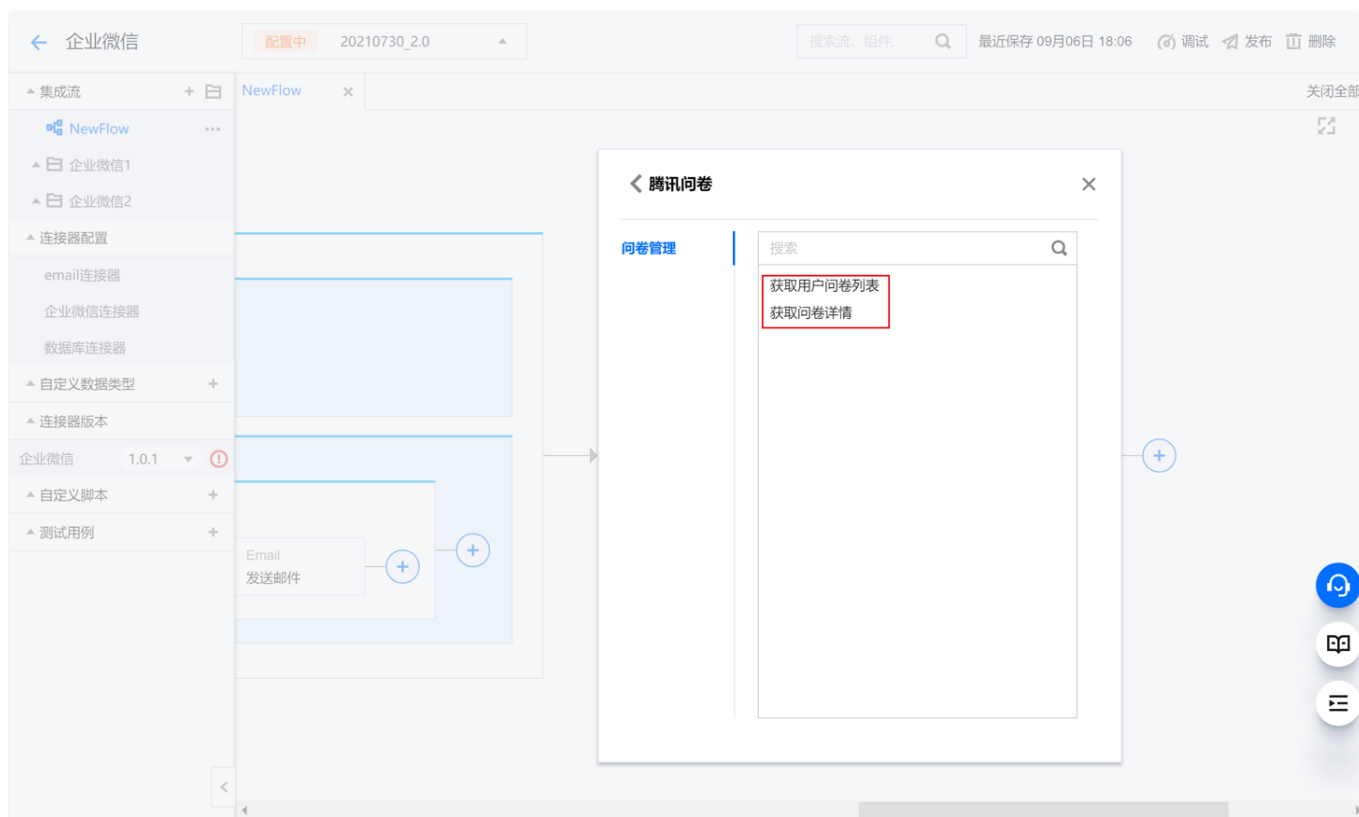
逻辑组件操作指引

对于首次创建的集成流，需要选择 Trigger 组件，单击 + 选择执行组件。若选择逻辑组件会直接打开该组件的配置页面。根据页面提示，进行通用配置，若有特殊需求，可选择行填写高级配置。



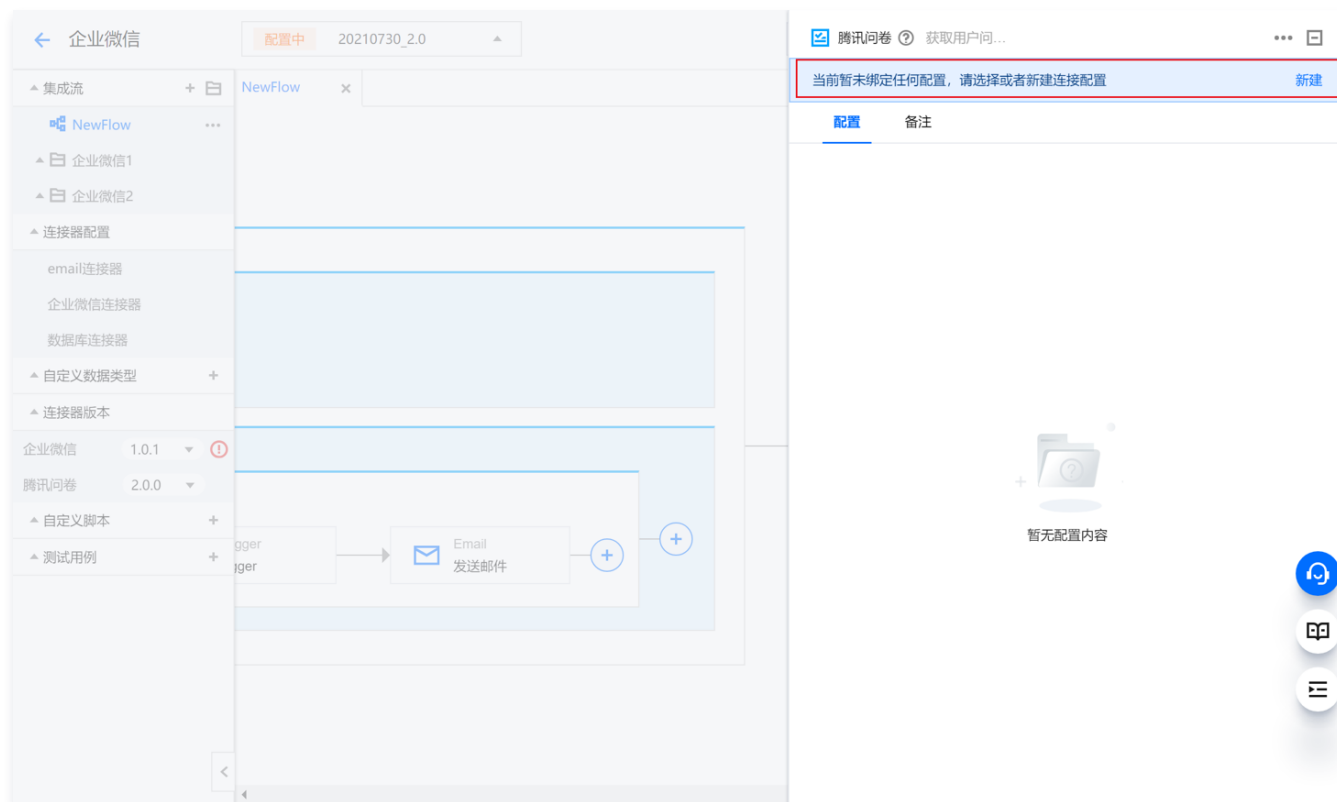
通用连接器或应用连接器操作指引

对于首次创建的集成流，需要选择 Trigger 组件，再单击 + 选择执行组件。若选择通用或应用连接器需要先选择需要的相关操作，再根据页面提示，进行 [连接器配置](#)，再填写通用配置，若有特殊需求，可选择行填写高级配置。

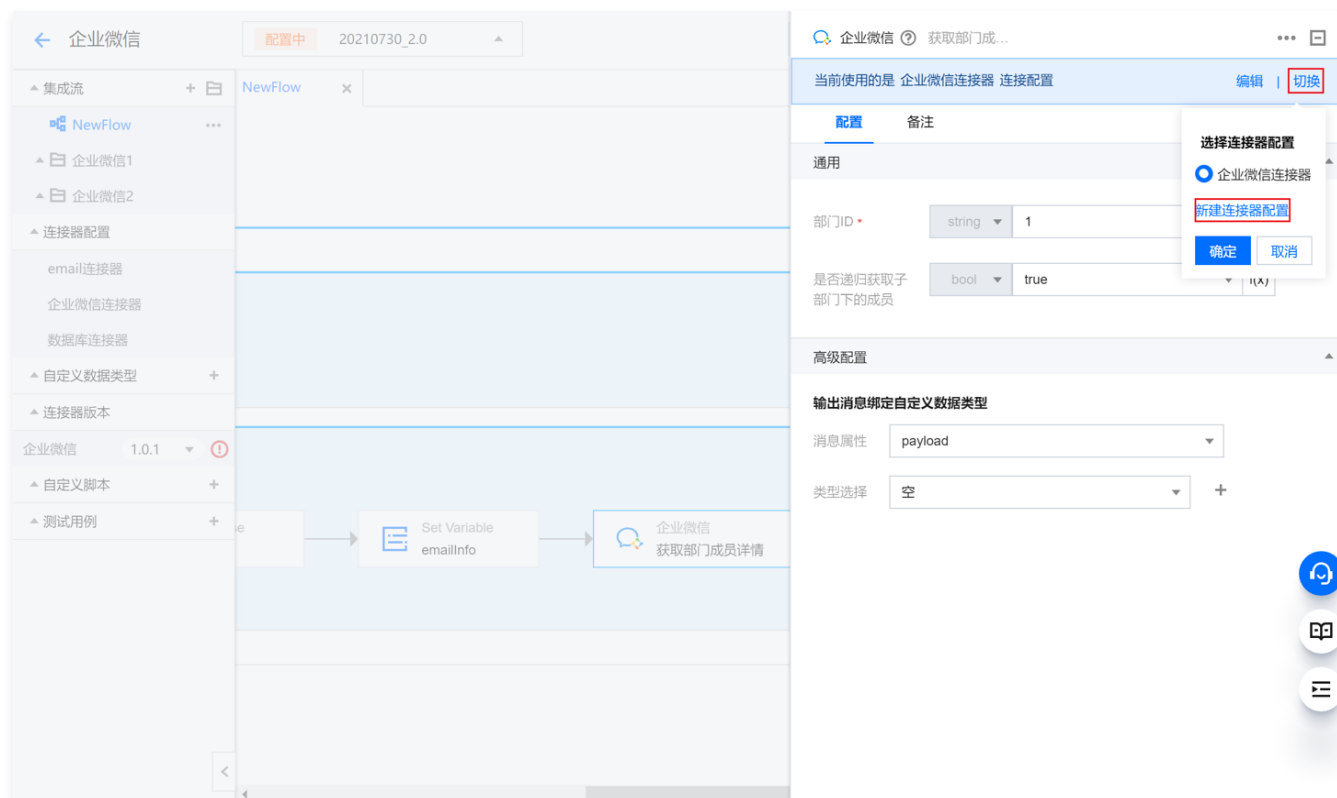


连接器配置

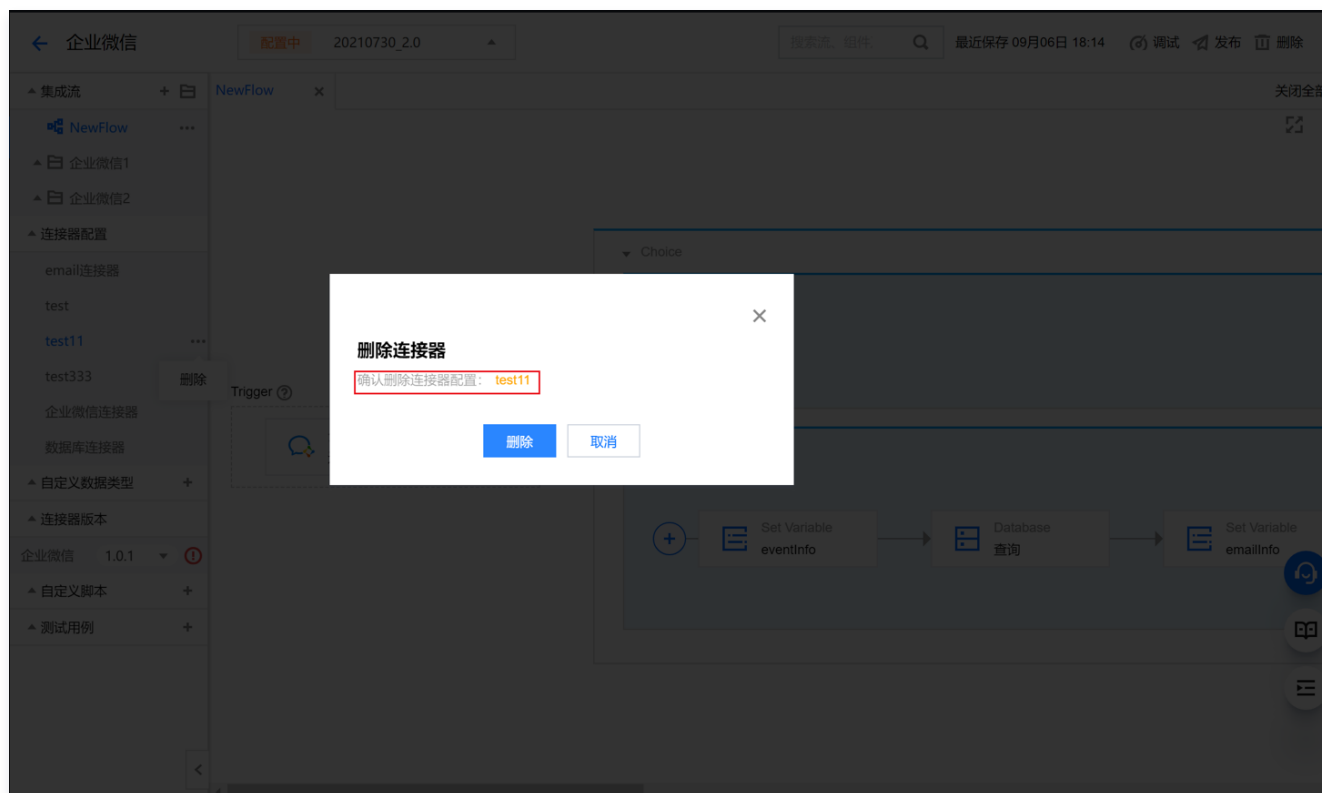
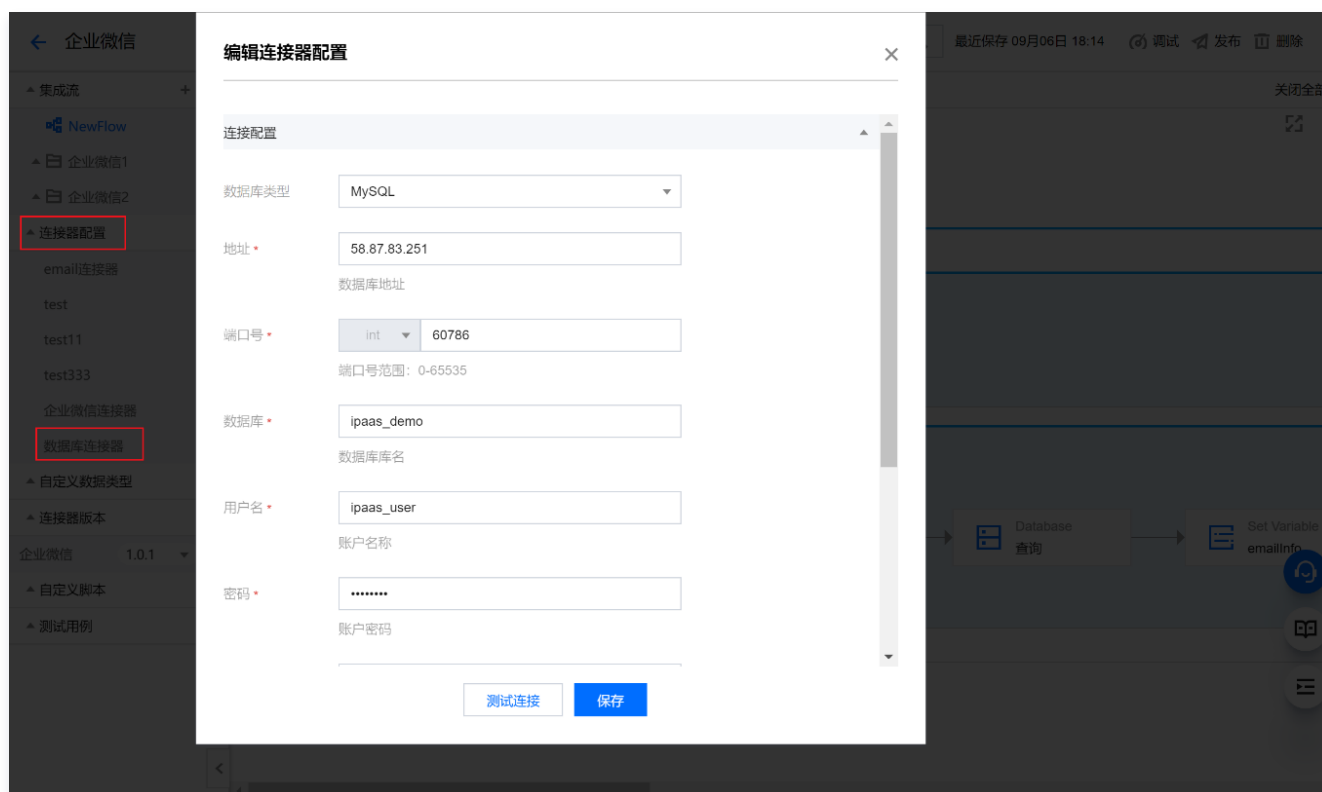
- **新建连接器配置**：选择连接器并在画布上创建连接器后，在弹出的配置页面左上角提示：当前暂未绑定任何配置，请选择或者新建连接配置。单击**新建**按照页面提示填写信息即可。



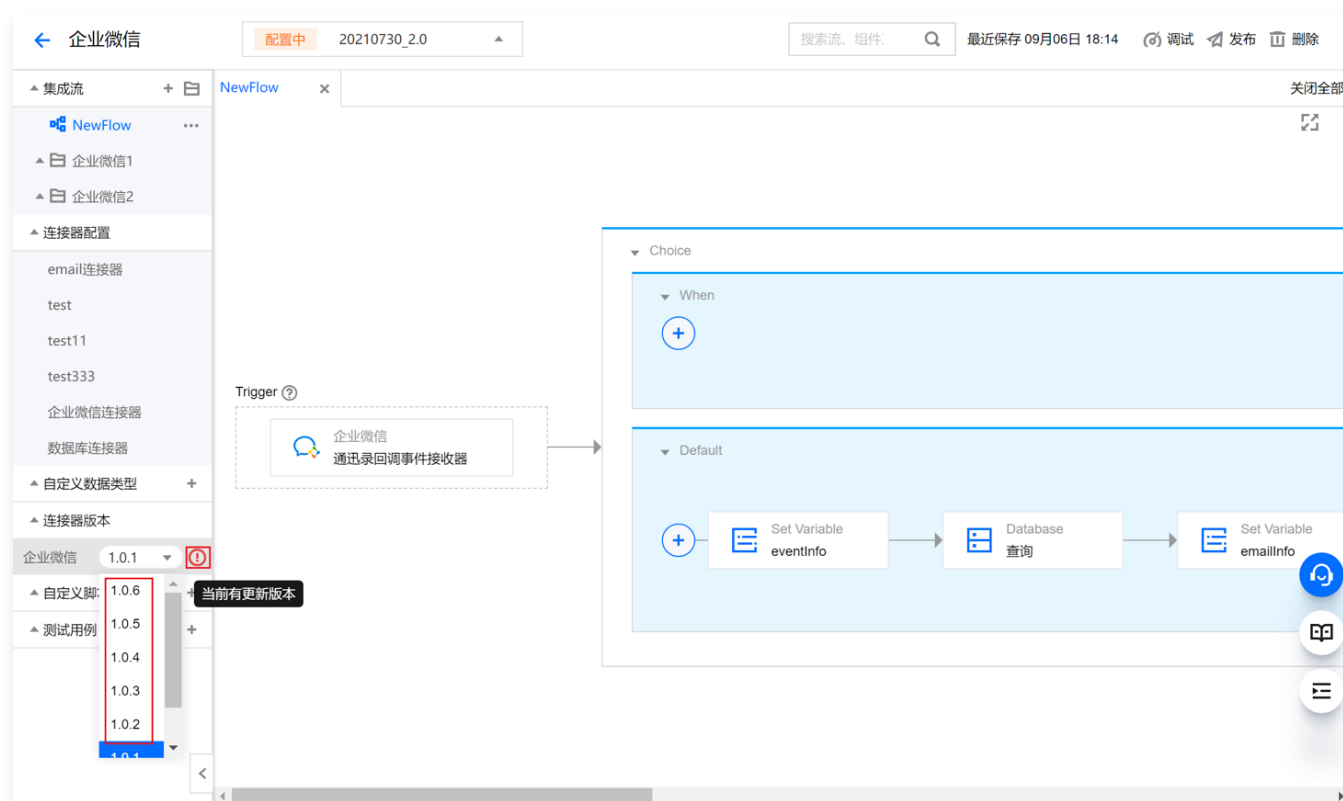
- **选择已有连接器配置**：对于非首次创建集成流，若同一应用中已有连接器配置，则客户可以自行选择已有配置，若客户想修改第三方凭证信息，可以单击右上角**切换连接器配置**，选择**新建**或**切换**已有连接器配置。



- **连接器配置**：对于已创建成功的连接器的连接配置，会在左侧**连接器配置**中展示。您可以直接单击通用配置下的内容进行修改，单击 ... 进行删除。



- 切换连接器版本: 对于已创建成功的应用连接器, 若连接器出现红色感叹号标志, 则表示现在使用的不是最新版本的连接器。连接器存在多版本, 可在左侧连接器版本中自行切换所需版本。

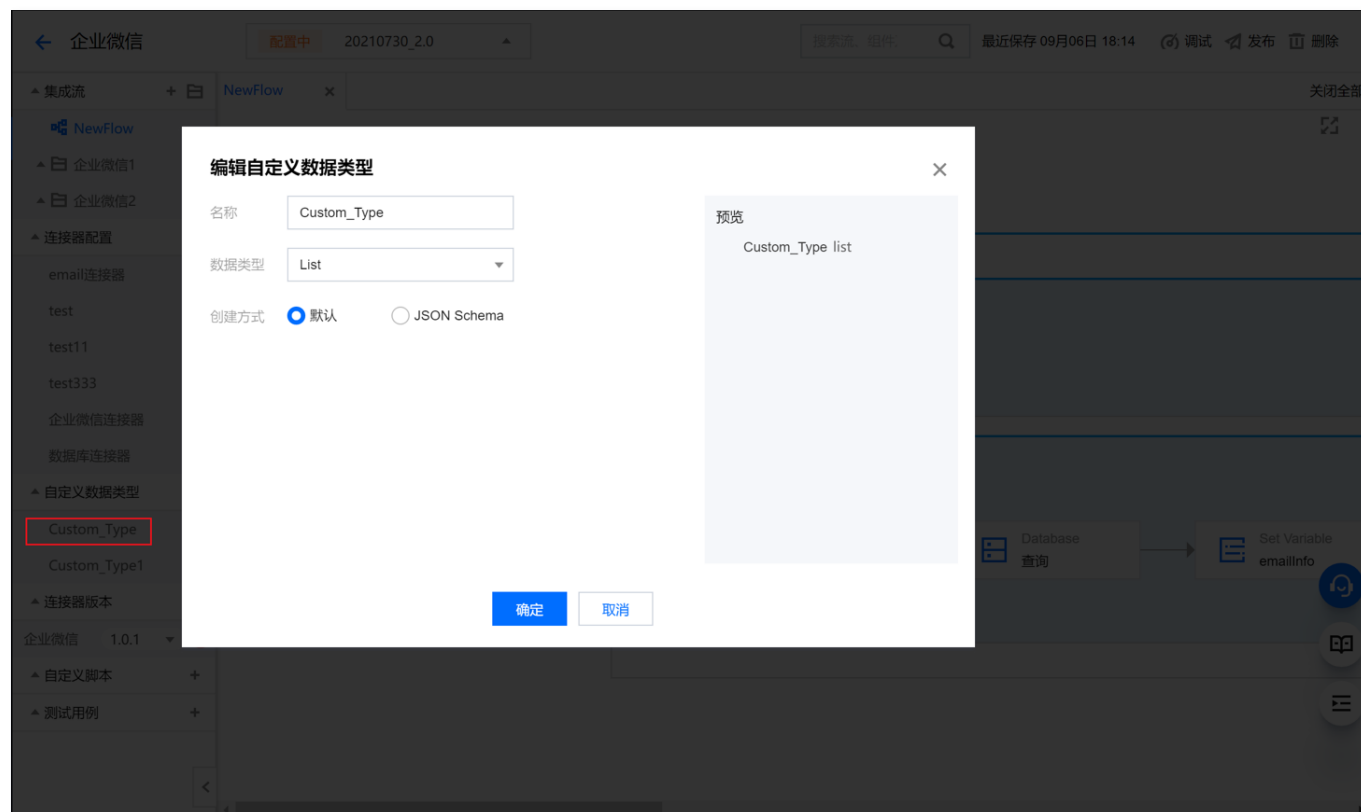


自定义数据类型

每个组件都有输入或者输出，自定义数据类型描述了一个组件的输入或输出的类型和格式。当给一个组件指定一个确定的自定义数据类型后，可以方便用户进行集成流的设计。自定义数据类型是在设计时绑定到一个组件上方便进行集成流的开发。当集成流在运行时，组件的输入和输出有可能和绑定的自定义数据类型不一致，但这不影响集成流的运行。自定义数据类型支持创建 List、Dict 和 Entity 类型，其中 Entity 类型包含 XML 和 JSON。List 和 Dict 数据类型可通过默认或 JSON Schema 创建，JSON Schema 支持自定义 List 或 Dict 中的每一项。

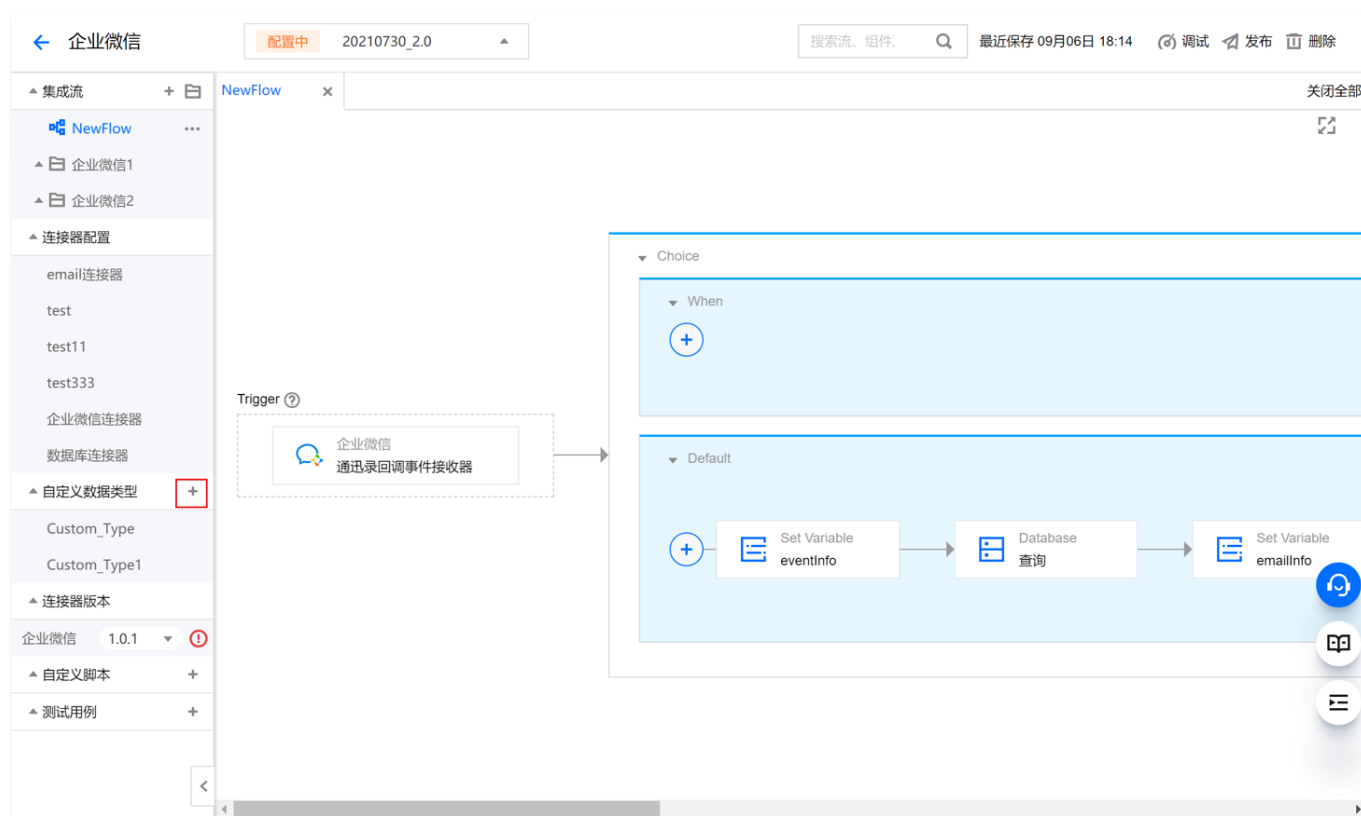
修改自定义数据类型

在应用配置界面，单击左侧导航栏中的自定义数据，会展示已有的自定义数据类型。若要修改自定义数据类型，单击对应数据类型名称即可。



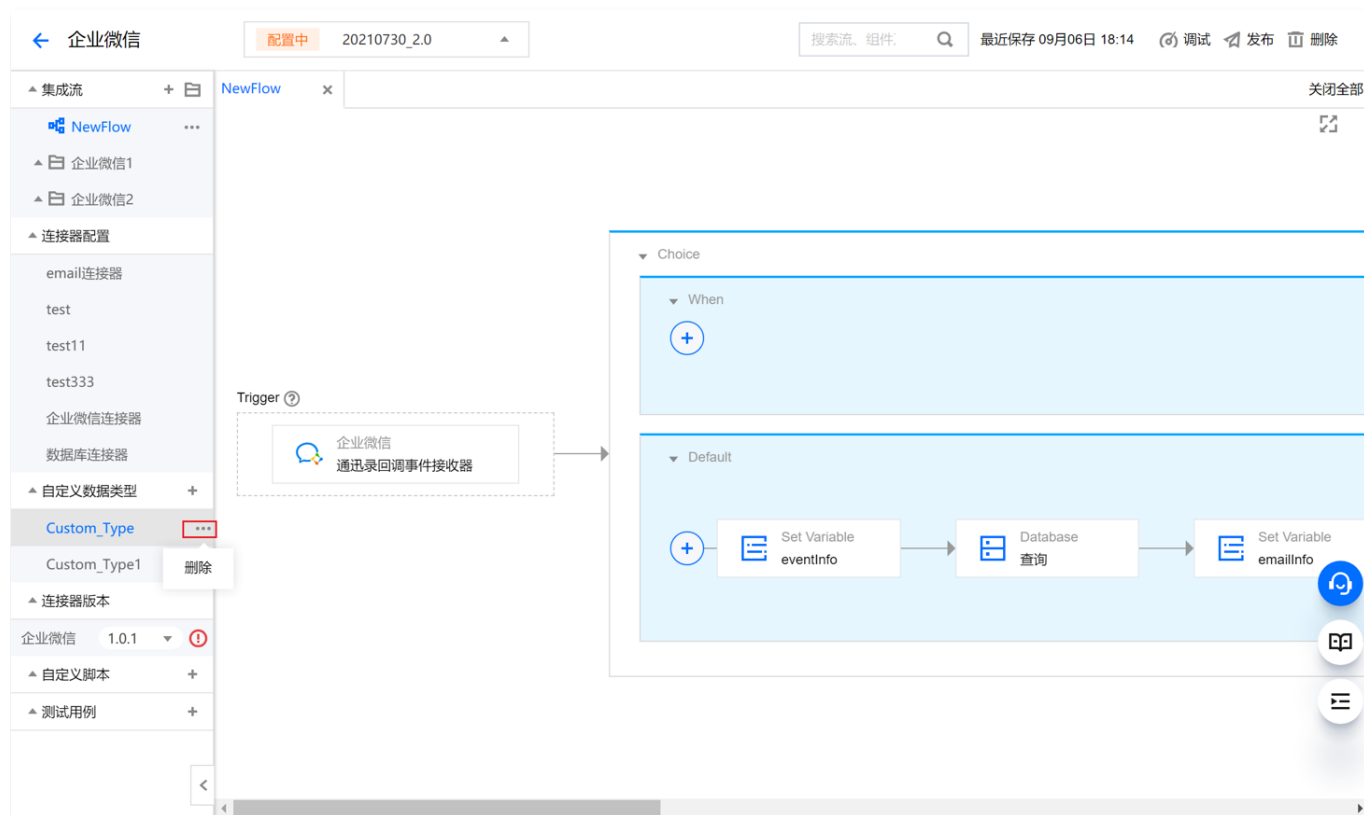
新增自定义数据类型

若要新增自定义数据类型，单击自定义数据类型右侧的 + 即可。



删除自定义数据类型

若要删除自定义数据类型，鼠标放在对应自定义数据名称上，单击 ... 选择删除即可，删除后不可恢复请谨慎操作。

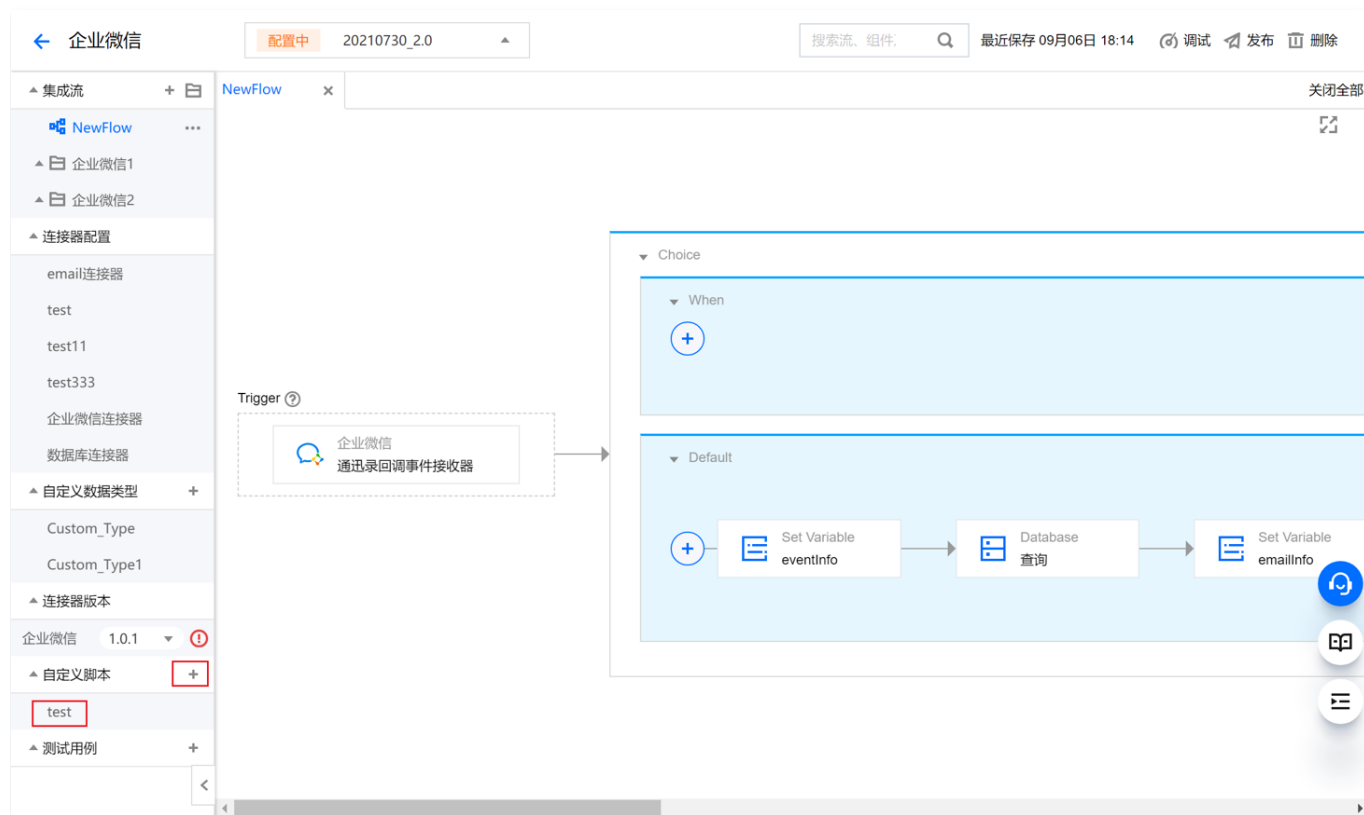


自定义脚本

平台侧提供了 Dataway 支持自定义脚本的功能，支持用户编写自定义的函数和组件。

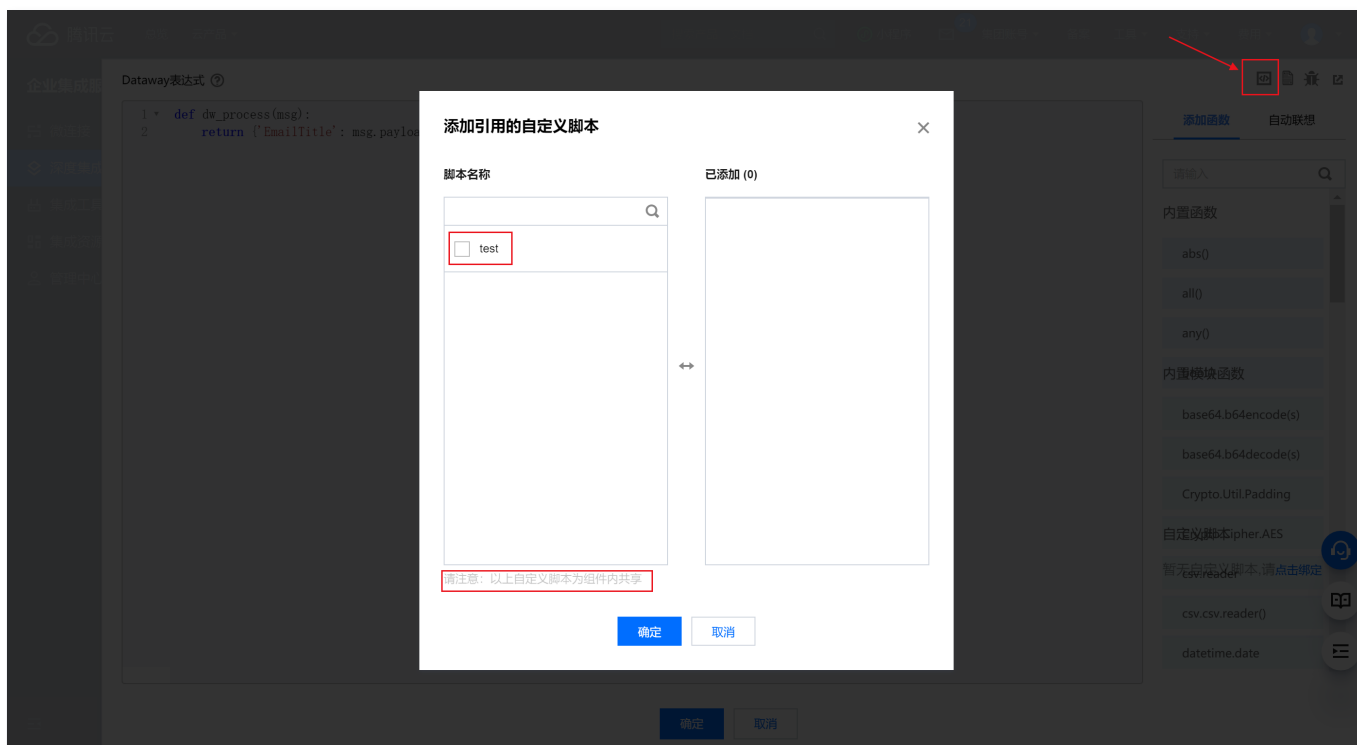
添加自定义脚本

若要添加自定义脚本，单击自定义脚本右侧的 + 即可，单击脚本名称可查看或修改自定义脚本。

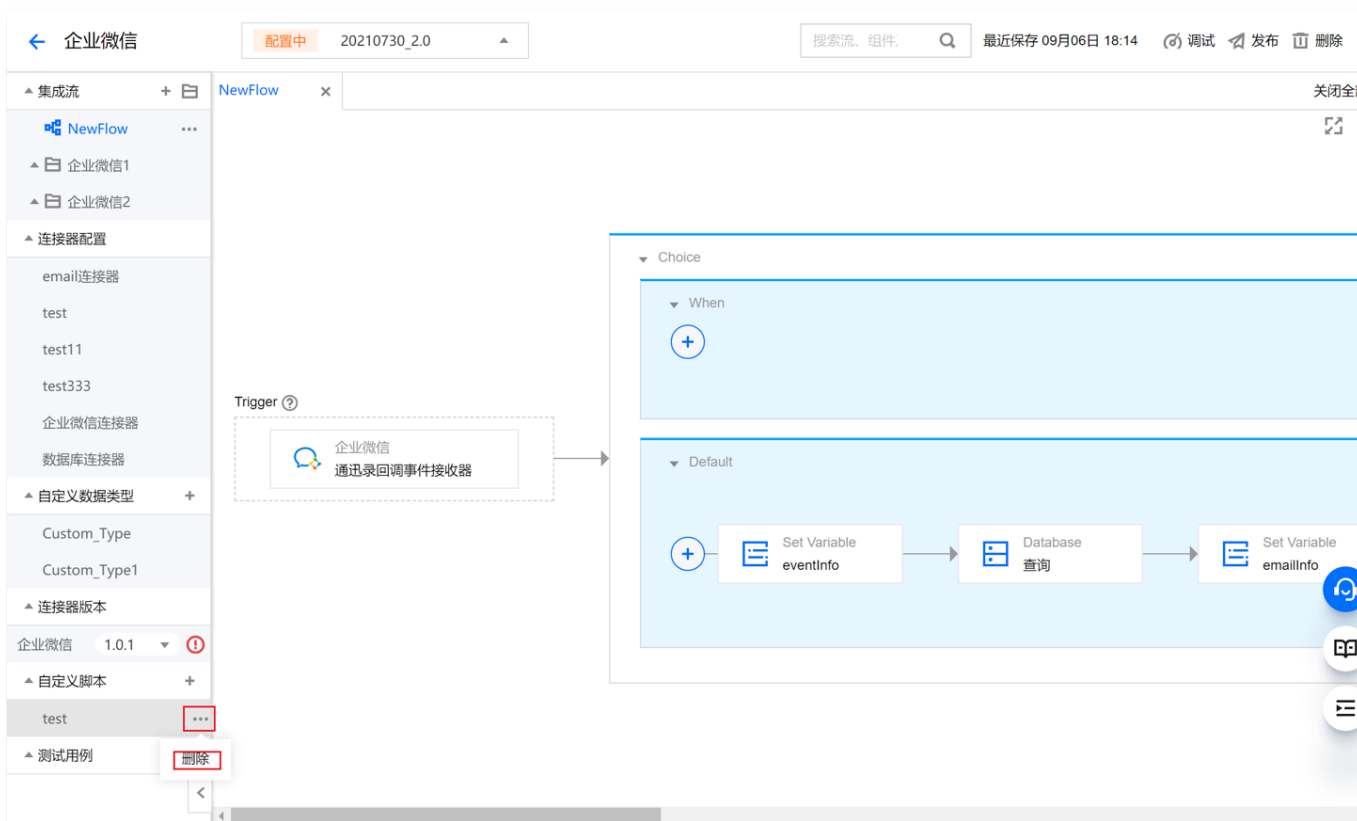


引用自定义脚本

选择组件单击 Dataway 表达式进行配置时，单击右上角的“</>”图标即可引用自定义脚本。



删除自定义脚本



测试用例

测试用例概念

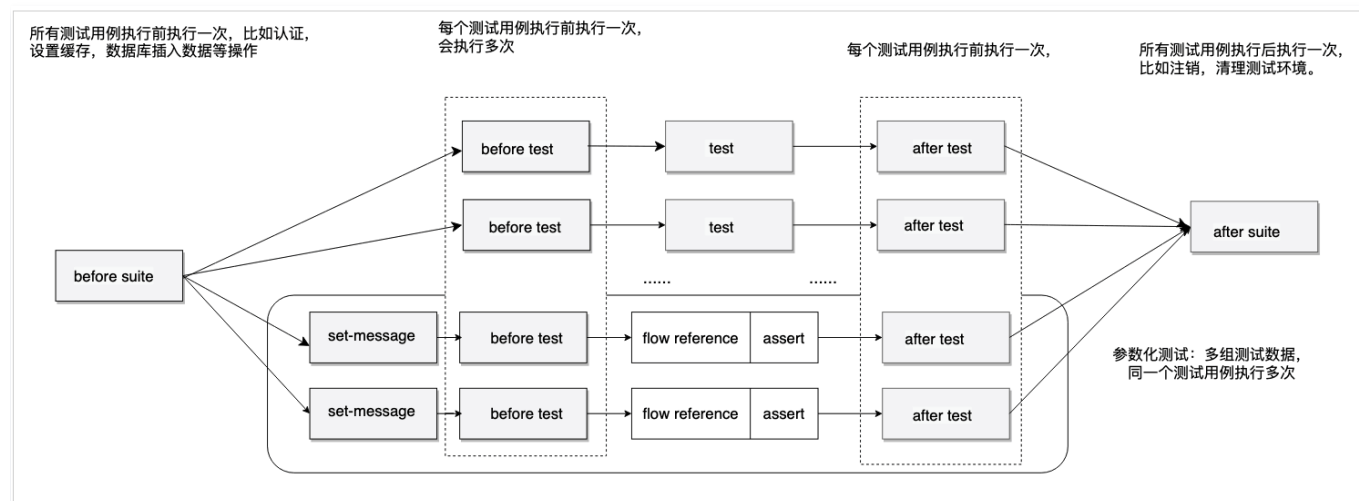
平台提供可视化的集成测试配置方案，方便您在单元测试的基础上，将所有模块按照设计要求组装成为完整的端到端流程，进行整体的集成测试。腾讯云数据连接器测试用例是以集成流为最小测试单位，通过验证集成流的运行结果来判断集成逻辑是否正确。通过测试用例，您可以减少开发测试投入的工时、加速开发迭代速度、增加测试工作自动化范围。

相关概念

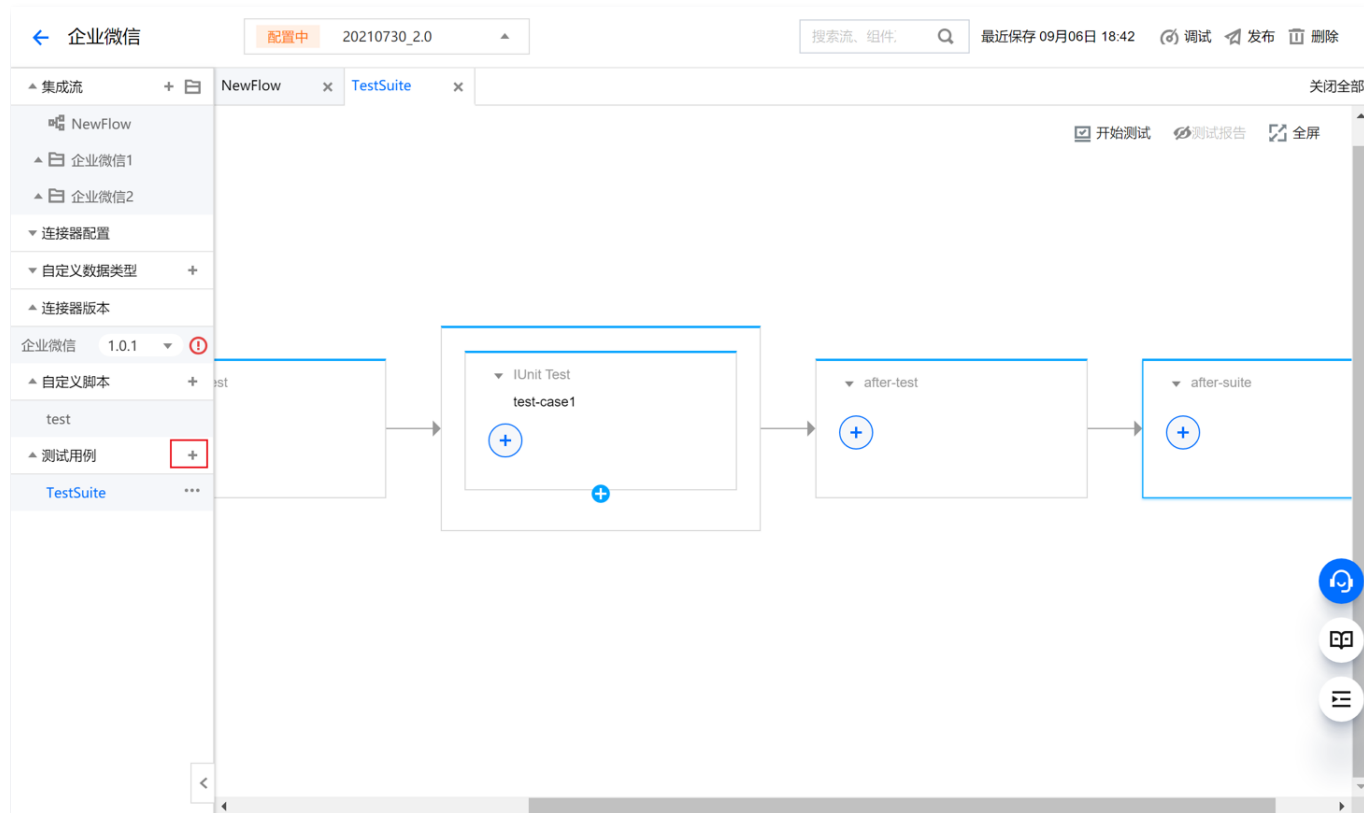
- **suite（测试套件）**：通常把一组相关的测试称为一个测试套件，是多个测试用例的集合，测试套件包括多个测试用例，一般也会有针对测试用例及其测试目的的详细说明，在进行测试时的系统组态资讯以及测试前需进行的步骤。在腾讯云数据连接器中，您可以对一个测试套件执行前（before suite）和执行后（after suite）的行为进行配置。
- **test case（测试用例）**：测试用例可用来测试您的集成流是否正确工作。
- **测试报告**：通过测试报告，您可以查看最近一次测试用例的执行情况，测试报告中包括测试结果和测试组件的覆盖率。

配置测试用例

与其他测试框架类似，平台提供多维度的测试配置，方便用户在测试用例和测试套件执行前后做一些操作。例如：在 before test 中配置一些在测试用例中会消费的消息、缓存数据等。多个测试用例之间不存在执行的先后顺序，腾讯云数据连接器将会通过并行的方式执行所有测试用例，提高测试效率。每一个测试组件都可以进行重命名和删除操作，且一个测试用例中至少要有一个 test 组件。



在测试用例配置完成后，您可以通过画布右上角的“开始测试”开启测试流程。系统会自动运行您配置的流程，在此期间您可以进行其他操作，待测试完成后，系统会在右上角进行提示，届时您可以回到“测试用例”界面，通过“测试报告”获取最近一次测试的结果。



测试报告中包括您本次测试的开始和结束时间、测试用例整体的执行情况、测试用例中集成流总测试覆盖率以及每条被测试的集成流的通过结果。同时您可以通过“下载报告”功能将报告保存在本地。

iUnit 测试用例报告

开始测试时间: 2021-09-06 18:47:18 测试结束时间: 2021-09-06 18:47:18

成功: 1 项 失败: 0 项 跳过: 0 项

测试用例名称	测试结果	耗时	原因
test-case1	成功	19ms	-

集成流总测试覆盖率: 0%

测试结果: 通过0项; 警告0项; 未通过1项;

集成流名称	测试状态	测试覆盖率
NewFlow	未通过	0%

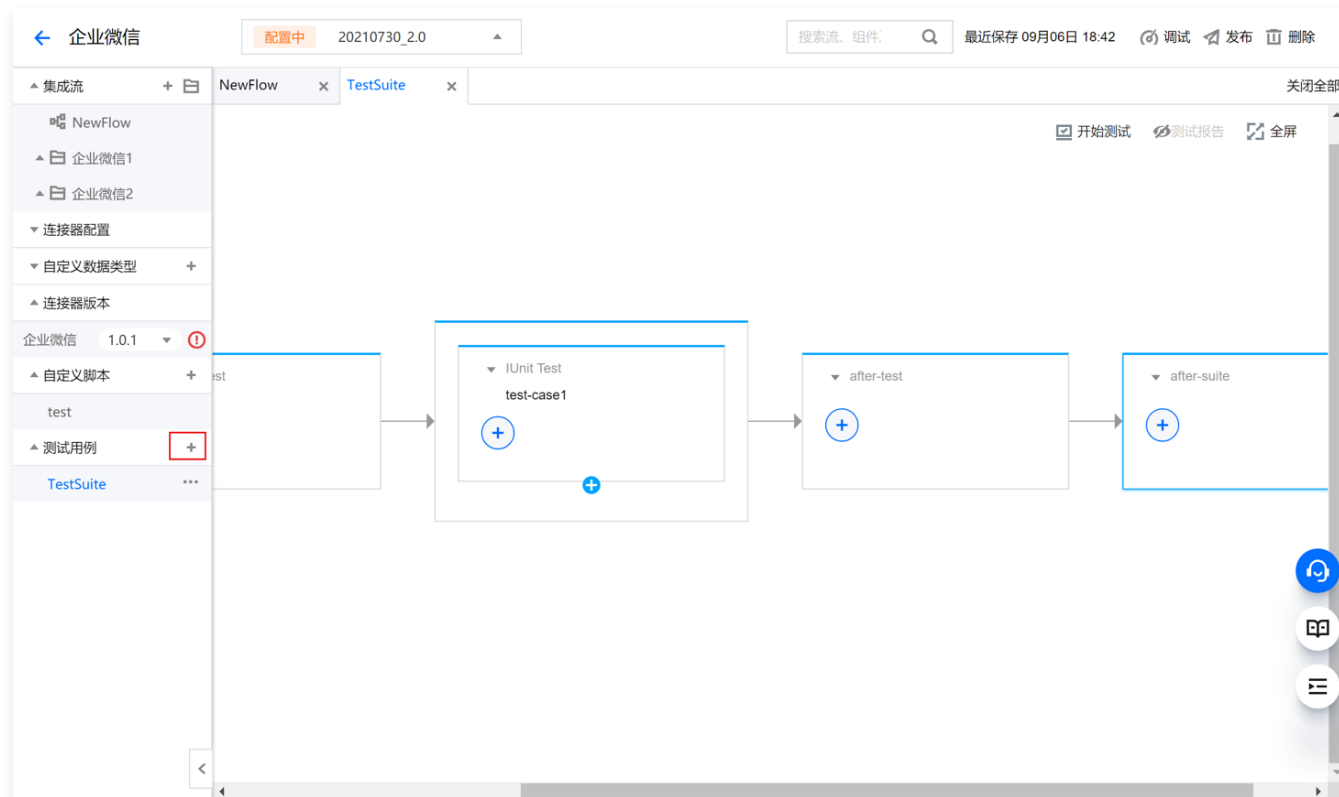
注: 集成流测试中, 通过(测试覆盖率为100%); 警告(测试覆盖率为50%-99%之间); 未通过(测试覆盖率低于50%)。

[下载报告](#) [取消](#)

相关概念

- **before suite**: 指在所有测试用例执行之前执行一次, 如认证之类的操作可以在此处配置。
- **before test**: 在每个测试用例执行前执行一次, 如果有多个测试用例, 则会被多次执行。
- **test**: 指具体的测试用例, 包括一条测试流。与普通的流配置方式类似, 用户可以在此编辑测试流, 指向需要测试的组件或者集成流。使用的过程中可以把测试数据组织起来, 用不同的测试数据调用相同的测试方法, 通过此种参数化测试的方式, 允许用户免去重复配置的工作, 而获取多组测试结果。同时, 测试的过程中可以通过 **skip** 功能跳过某个不希望运行的组件或者步骤。
- **after test**: 在每个测试用例执行后执行一次。如果有多个测试用例, 将会被多次执行。

- **after suite**: 在所有测试用例执行后执行一次。您可以在此处配置一些完成动作，如用户注销等行为。



多人协作开发模式

腾讯云数据连接器提供**多人协作开发**特性，让用户团队内多名成员可以同时开发同一个应用，提升开发效率。且通过设置编辑时加锁机制，解决了多人同时编辑带来的版本不一致问题。

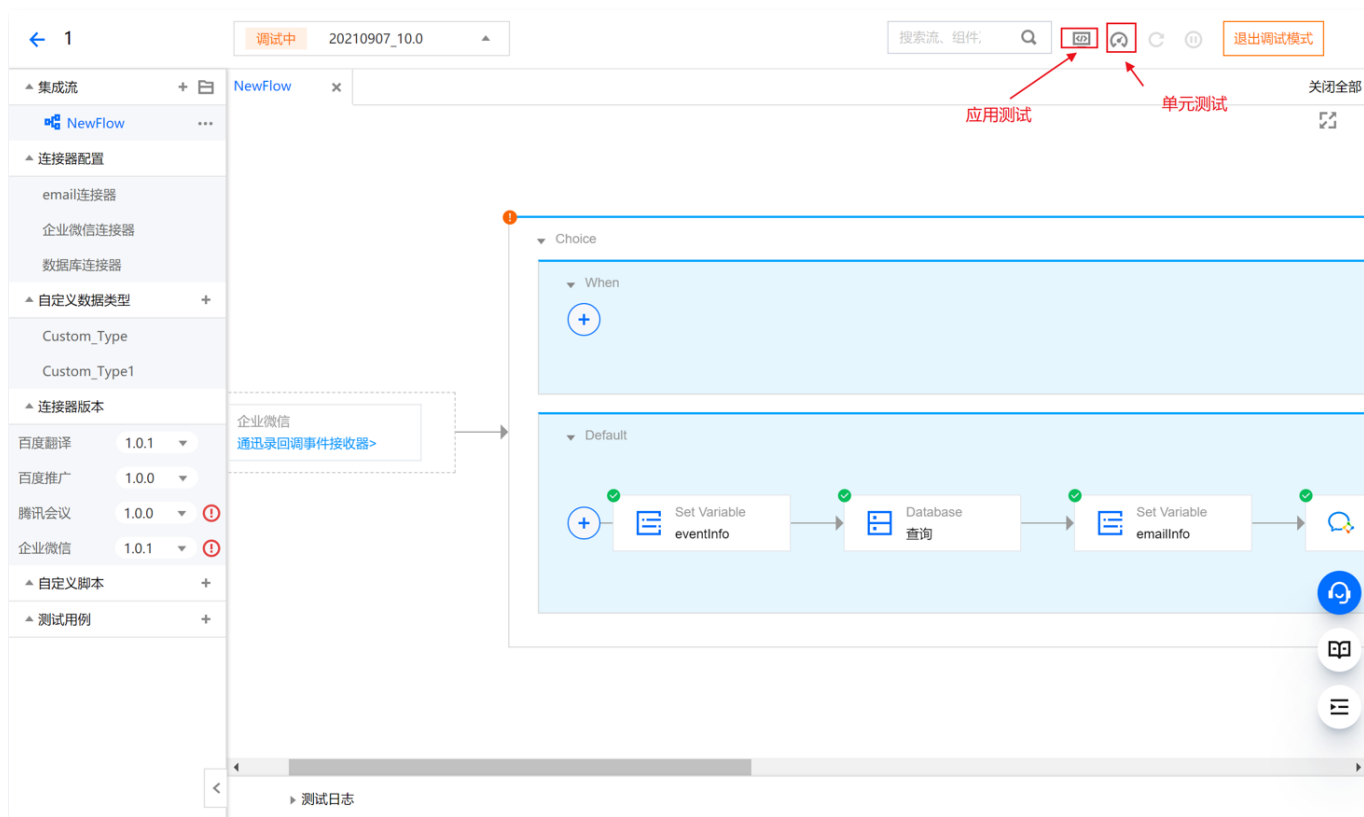
⚠ 注意

多人开发是指支持单个应用下多个用户同时开发多条集成流。单条集成流在同一时间只支持被一位成员编辑，其余成员点击此集成流会处于锁定状态（不可编辑）。避免多人同时编辑带来的版本不一致问题。



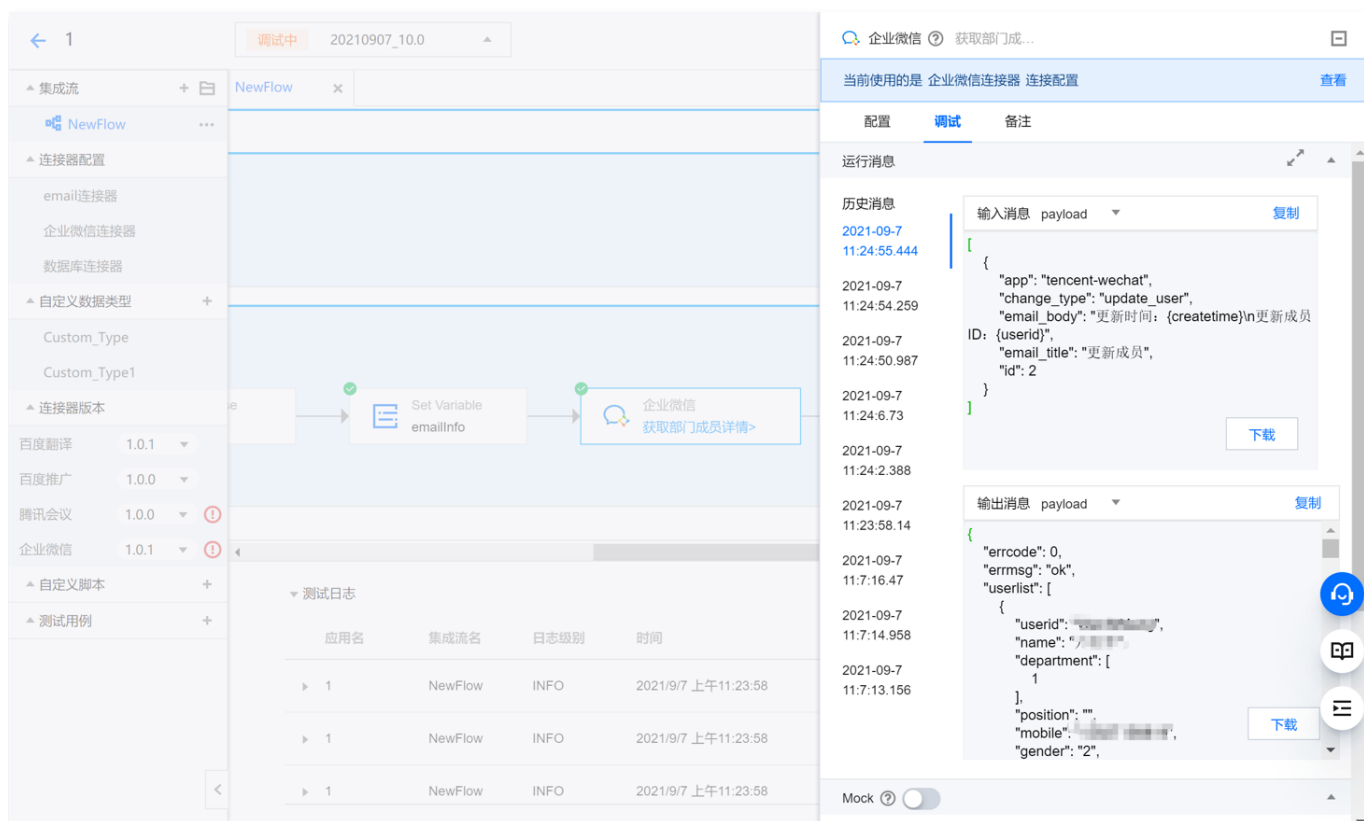
调试模式

腾讯云数据连接器在集成应用开发的过程中提供丰富的调试能力，方便用户在发布前提前测试应用的可用性。单点击**调试**进入调试模式，支持应用测试、单元测试、查看节点输入输出等能力。



单个组件调试内容查看

调试模式下，每个组件打开后新增调试标签，可以查看每个组件的输入输出，也可以模拟该组件的输入数据辅助调试。



应用测试

应用测试时，模拟一个触发操作对当前集成应用做整体测试，将集成应用发布到沙箱环境中运行，对应触发链接可直接复制 Trigger 后的链接地址。触发成功后画布底部会打印对应的测试日志。

应用名	集成流名	日志级别	时间	日志数据
1	NewFlow	INFO	2021/9/7 下午12:19:08	{ "category": "logger", "componentId": "8", "date": "2021-09-07 12:19:08.911", "docId": "f..." }
1	NewFlow	INFO	2021/9/7 下午12:19:07	{ "category": "logger", "componentId": "8", "date": "2021-09-07 12:19:07.990", "docId": "f..." }
1	NewFlow	INFO	2021/9/7 下午12:19:11	{ "category": "http.listener", "componentId": "...", "date": "2021-09-07 12:19:11.337", "docId": "f..." }

单元测试

单元测试支持对一个或相邻的几个组件进行局部测试，在流比较复杂的情况下，做整体应用测试的耗时可能较长，可以做局部测试来减少耗时且尽快定位哪些组件可能异常（集成流首位的触发组件不可以被选中）。

单元测试

1 请选择所要测试的一个或多个组件 > 2 请模拟单元测试输入数据

取消 下一步

选中单元测试的节点后，需要模拟单元测试的输入数据，如果之前已经进行过整体应用测试，平台会自动拉取历史测试数据作为输入来进行测试，若之前不存在历史测

单元测试

1 请选择所要测试的一个或多个组件

2 请模拟单元测试输入数据

☒ 使用首个组件的历史输入数据作为单元测试的输入数据

历史消息

2021-09-7 12:19:36.983

payload 值

entity.json

{"ToUserName": "v...7.

2021-09-7 12:19:35.775

attribute 值

变量名

变量值

操作

2021-09-7 12:19:32.928

cookies

dict

编辑

删除

2021-09-7 12:19:14.996

headers

multimap

编辑

删除

2021-09-7 12:19:11.339

host

string

删除

method

string

POST

删除

上一步

开始测试

取消

配置调试好集成应用后，先退出调试模式再进行发布，可以将其进行多地域发布，来减少由于环境导致的集成不可用的情况。

目前支持天津和广州地域的共享集成，其他地域敬请关注。

版本管理

第33 共732页

说明

2022年9月20号后创建的应用新增展示发布环境字段，再次发布时保留上次所选发布环境，方便管理。

← 我是测试应用2

运行中

20220907_1.0

① 运行中的版本无法修改。点击右侧「复制」后在新版本中进行操作

搜索流、组件

最近保存 09月15日 14:40

停止

复制

集成流

NewFlow

自定义数据类型

连接器版本

自定义脚本

测试用例

运行中

20220907_1.0 天津

更新于 3 小时前

暂无更多数据

关闭全

Trigger

Scheduler

scheduler

+

逻辑组件使用指南

Async

最近更新时间：2023-06-08 14:47:54

简介

Async 属于流程控制类组件，在该组件中，可以配置子流执行异步任务，类似于开辟线程。对于写文件、写数据库等任务，可以放置在 Async 中执行。子流在执行时，父流中的 message 会全部传递到 Async 的子流中，子流在执行完成后，结果不会传递给父流，同时，子流在报错时，父流也不会感知，因此需要在子流中进行错误处理。所以，Async 多适合执行一些写文件等耗时操作。

操作说明

参数配置

无

输入到子流中的 message

message 属性	值
payload	继承 Async 上一个组件的 payload
error	空
attribute	继承 Async 上一个组件的 attribute 信息
variable	继承 Async 上一个组件的 variable 信息

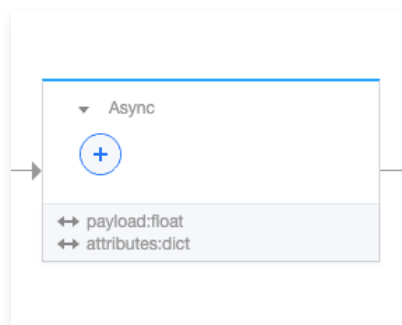
输出

组件输出的 message 信息如下：

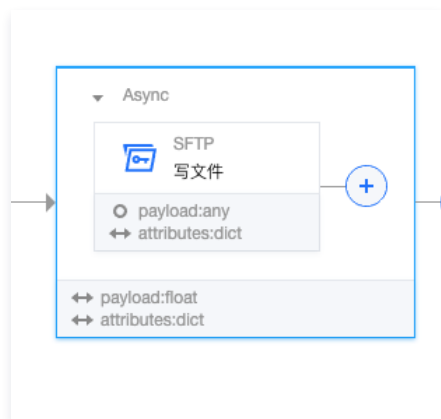
message 属性	值
payload	继承 Async 上一个组件的 payload
error	空
attribute	继承 Async 上一个组件的 attribute 信息
variable	继承 Async 上一个组件的 variable 信息

案例

1. 添加 Async 组件。



2. 使用 SFTP 组件，向目标服务器写入数据。当执行到 Async 时，系统会异步执行 Async 中的子流。



Break

最近更新时间：2023-06-08 14:47:54

简介

Break 组件需要和 For Each 或者 While 组件搭配使用，用来中断循环，即使循环条件没有 False 或者序列还未递归完成，也会停止执行循环语句。当存在嵌套循环时，Break 将跳出所在层的循环，并开始执行下一行代码。

操作说明

参数配置

无

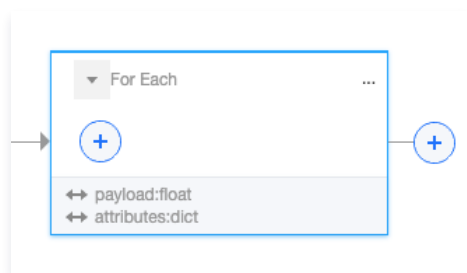
输出

组件输出的 message 信息如下：

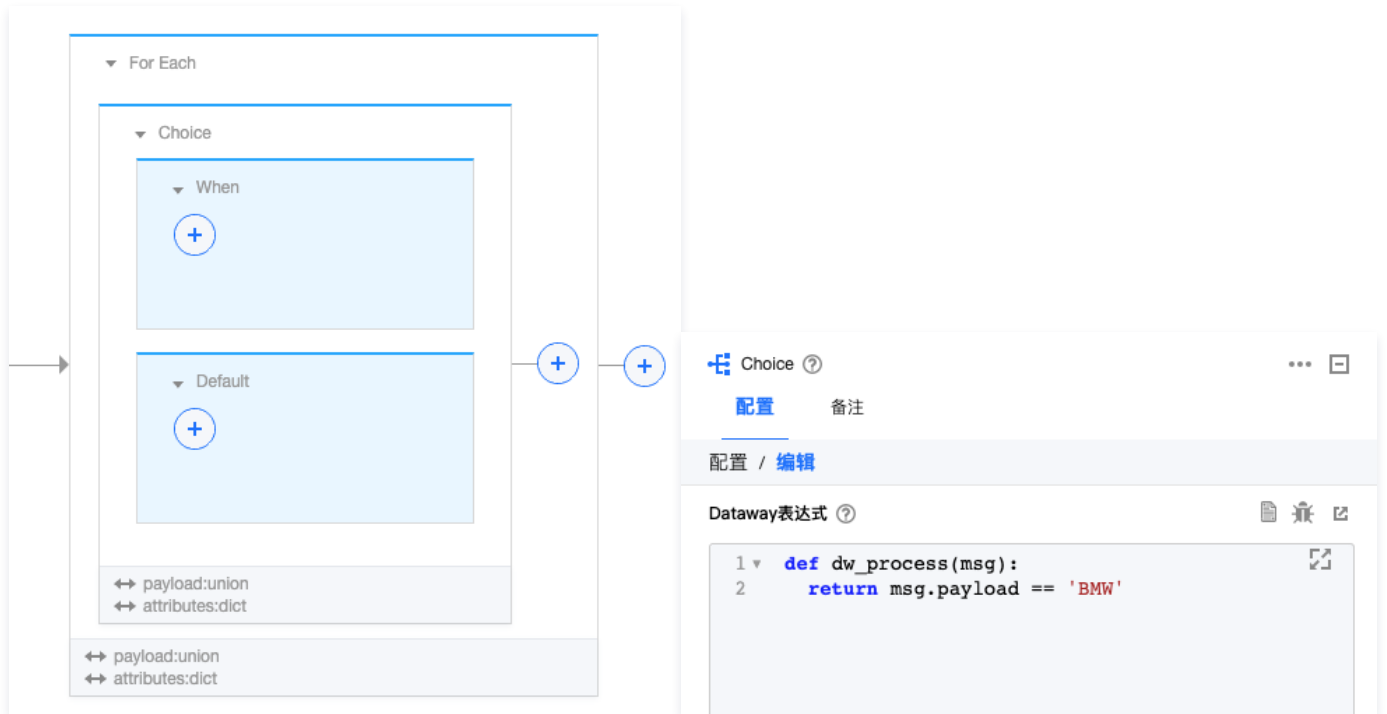
message 属性	值
payload	当在 For Each 组件中使用时，Break 跳出循环后，payload 中的数据继承自 For Each 的上一个组件的 payload；当在 While 组件中使用时，跳出循环后，payload 的数据继承自 Break 的上一个组件输出的 payload。
error	空。
attribute	在 For Each 组件中使用时，Break 跳出循环后，attribute 继承自 For Each 的上一个组件输出的 attribute；在 While 组件中使用时，Break 跳出循环后，attribute 继承自 While 的上一个组件输出的 attribute。
variable	在 For Each 中使用时，Break 跳出循环后，variable 中的变量是 For Each 的上一个组件输出的 variable 变量加上 For Each 子流中声明的 variable 变量；在 While 组件中使用时，variable 中的变量是 While 的上一个组件输出的 variable 变量加上 For Each 子流中声明的 variable 变量。

案例

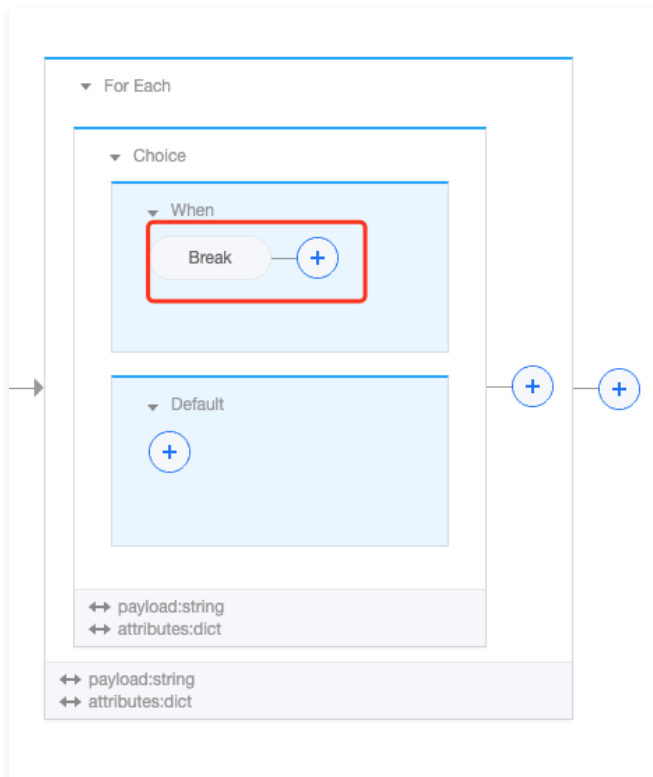
1. 添加 For Each 组件，设置要遍历的列表。



2. 使用 Choice 组件，并在 When 节点中添加判断条件。



3. 在 When 节点中添加 Break 组件，当遍历到 BMW 时，跳出 For Each 循环。



Cache

最近更新时间：2023-06-08 14:47:54

简介

Cache 组件用于缓存流运行中产生的中间数据，同一个应用中的不同流，可以共享 Cache 组件中缓存的数据。目前 Cache 组件只能缓存字符串类型数据，并对每个用户的使用空间进行限制，每个用户可存储的数据量最大为10MB，可存储的数据条数最大为1000条，单条数据大小不能超过100KB，数据的缓存时间最长为30天。存储数据量和数据条数，都支持动态配置，如果需要扩容，可联系管理员进行操作。

操作说明

存储

参数配置

参数	数据类型	描述	是否必填	默认值
键值	String	数据的唯一标识，根据该参数查找数据。	是	无
数据	String	待存储的数据。	是	无
存储时长	int	数据过期时间，取值范围为1~2592000，不支持永久保存。	是	无

cache ? 存储

配置 输入 输出 备注

通用

键值

string

key

唯一标识，可用来读取数据

数据

string

value

需要存储的数据

存储时长(秒)

string

3600

存储时长，取值范围为1~2592000

输出

组件输出的 message 信息如下：

message 属性	值
payload	当数据存储成功后，payload 中的值为“OK”；存储失败时，则值为“fail”。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	存储成功时，variable 中的信息与上个组件相同；存储失败时，variable 在上个组件的信息基础上增加变量“err”，“err”描述失败原因。

● 存储成功

当存储成功时，payload 中的值为“OK”。

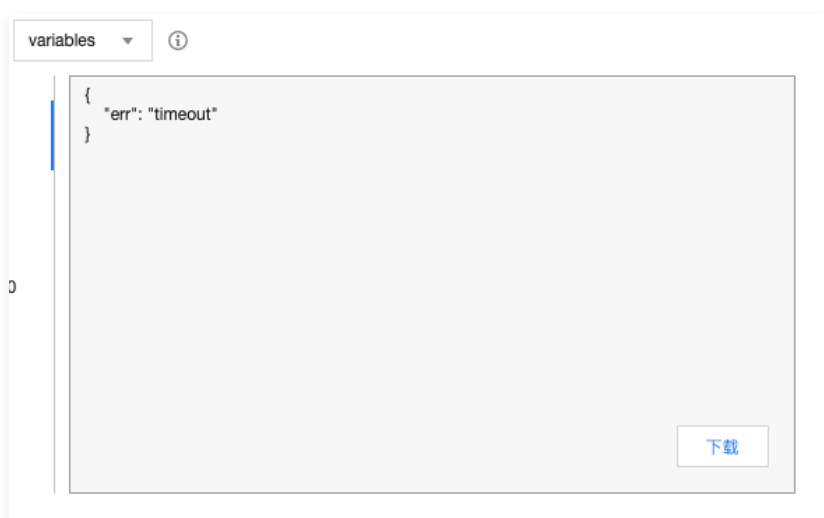


• 存储失败

当存储失败时，payload 中的值为“fail”。



存储失败时，变量“err”中包含失败的原因。



存储（仅在键不存在时，执行存储）

当选择该操作时，如果已经使用设置的键保存过数据，且该数据未过期，则会保存失败。

参数配置

参数	数据类型	描述	是否必填	默认值
键值	String	数据的唯一标识，根据该值索引数据，如果两个数据的键相同，则后执行的存储操作会覆盖之前的数据。	是	无
数据	String	待存储的数据。	是	无
存储时长	int	数据过期时间，取值范围为1~2592000，不支持永久保存。	是	无

配置界面

cache 存储(仅在... 配置 输入 输出 备注

通用

键值

stringkey

唯一标识，可用来读取数据

数据

stringvalue

需要存储的数据

存储时长(秒)

string3600

存储时长，取值范围为1~2592000

输出

组件输出的 message 信息如下：

message 属性	值
payload	当数据存储成功后，payload 中的值为 “OK”， 存储失败时，则值为 “fail”。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	存储成功时，variable 中的信息与上个组件相同；存储失败时，variable 在上个组件的信息基础上增加了变量 “err”，“err” 描述失败原因。

- 存储成功**
当设置的键不存在时，payload 输出 “OK”。

payload

1

"OK"

下载

• 存储失败

当存储失败时，payload 中的值为“fail”。

payload

"fail"

下载

存储失败时，变量“err”中可查看失败的原因。

variables

{

"err": "dataExist"

}

下载

读取

参数配置

参数	数据类型	描述	是否必填	默认值
键值	String	数据的唯一标识，根据该参数查找数据。	是	无

配置界面

cache ? 读取

配置 输入 输出 备注

...

通用

键值

string key

唯一标识，用来读取数据

输出

组件输出的 message 信息如下：

message 属性	值
payload	读取存在时，payload 保存待查找的数据；数据不存在时，payload 中为空字符串。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	读取成功时，variable 中的信息与上个组件相同；读取失败时，variable 在上个组件的信息基础上增加变量“err”，“err”描述失败原因。

- 数据存在
当要查找的数据存在时，payload 中会保存待查找的数据，使用 msg.payload 即可读取该数据。

payload

"value"

下载

• 数据不存在
当要查找的数据不存在时，payload 中返回空字符串。

payload

下载

删除

参数配置

参数	数据类型	描述	是否必填	默认值
键值	String	数据的唯一标识，删除该值标识的数据。	是	无

配置界面

cache ? 删除

配置 输入 输出 备注

...

☐

通用

键值

string

key

唯一标识，用来查找数据

输出

组件输出的 message 信息如下：

message 属性	值
payload	当数据删除成功后，payload 中的值为 “deleted” ；当待删除的数据不存在时，payload 中的数据为 “data not exist” 。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	删除成功时，variable 中的信息与上个组件相同；删除失败时，variable 在上个组件的信息基础上增加变量 “err” ， “err” 描述失败原因。

删除成功

删除成功后，payload 中会返回 “deleted” 。

payload

?

"deleted"

下载

删除失败

当删除一个不存在的数据时，payload 中的值为 “data not exist” 。

payload

"data not exist"

下载

检查是否存在

参数配置

参数	数据类型	描述	是否必填	默认值
键值	String	数据的唯一标识，查找该值对应的数据。	是	无

配置界面

cache

检查是否存在

配置

输入

输出

备注

...

☰

通用

键值

string

key

唯一标识，用来查找数据

输出

组件输出的 message 信息如下：

message 属性	值
payload	当数据存在时，payload 中的数据为整数1；当数据不存在时，payload 中的数据为整数0。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。

variable	执行成功时，variable 中的信息与上个组件相同；执行失败时，variable 在上个组件的信息基础上增加变量“err”，“err”描述失败原因。
----------	---

• 数据存在

当要查找的数据存在时，payload 中的值为1。

payload

1

下载

• 数据不存在

当数据不存在时，payload 中的值为0。

payload

0

下载

案例

存储数据

1. 添加 Cache 组件。



2. 选择存储。



3. 填入键值、数据、存储时长。



读取数据

1. 选择读取。



2. 填入键值即可读到数据，下一个节点可以使用 msg.payload 访问获取到的数据。



Choice

最近更新时间：2023-06-08 14:47:55

简介

Choice 是分支选择语句，基于不同的条件执行不同的动作，类似于 if-else。Choice 下包含两类子节点，分别为 When 和 Default，在 Choice 中可以添加多个 When 节点，每个 When 节点中都包含一个 bool 表达式，Choice 组件会对 When 节点逐个匹配，直到表达式满足条件，则执行该 When 节点中配置的子流。当所有的 When 条件都无法匹配时，会执行 Default 分支的动作。

操作说明

参数配置

在 When 节点中，可以配置条件语句，用来控制分支选择。

参数	数据类型	描述	是否必填	默认值
条件	bool	条件判断，当条件满足时，执行对应的子流。	是	无

配置界面



输入到子流中的 message

message 属性	值
payload	继承 Choice 上一个组件的 payload。
error	空。
attribute	继承 Choice 上一个组件的 attribute 信息。
variable	继承 Choice 上一个组件的 variable 信息。

输出

Choice 属于流控类组件，输出的 message 是具体执行的子流生成的 message。

组件输出的 message 信息如下：

message 属性	值
payload	继承子流输出的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。

attribute	继承子流输出的 attribute。
variable	继承子流输出的 variable。

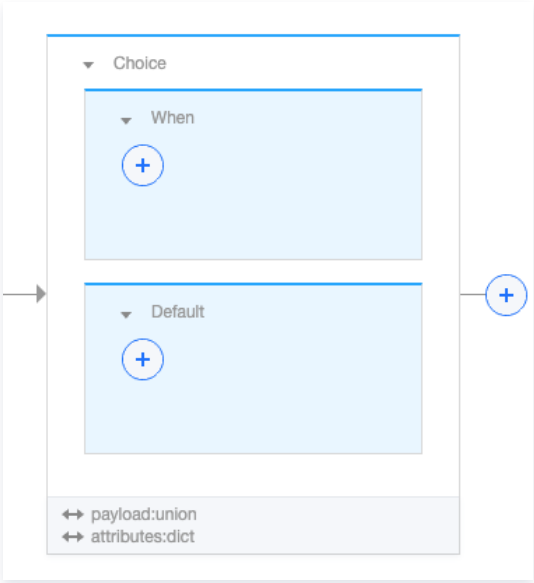
案例

在该案例中，我们将 score 映射为不同的 level。

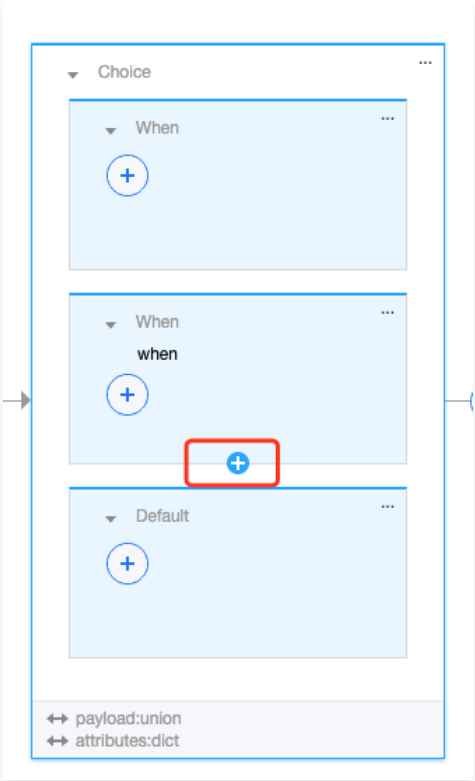
- 当 score 大于等于90时，level 为” A “。
- 当 score 大于等于80，小于90时，level 为” B “。
- 当 score 大于等于60，小于80时，level 为” C “。
- 当 score 小于60时，level 为”D” 。

操作步骤：

1. 添加 Choice 组件。



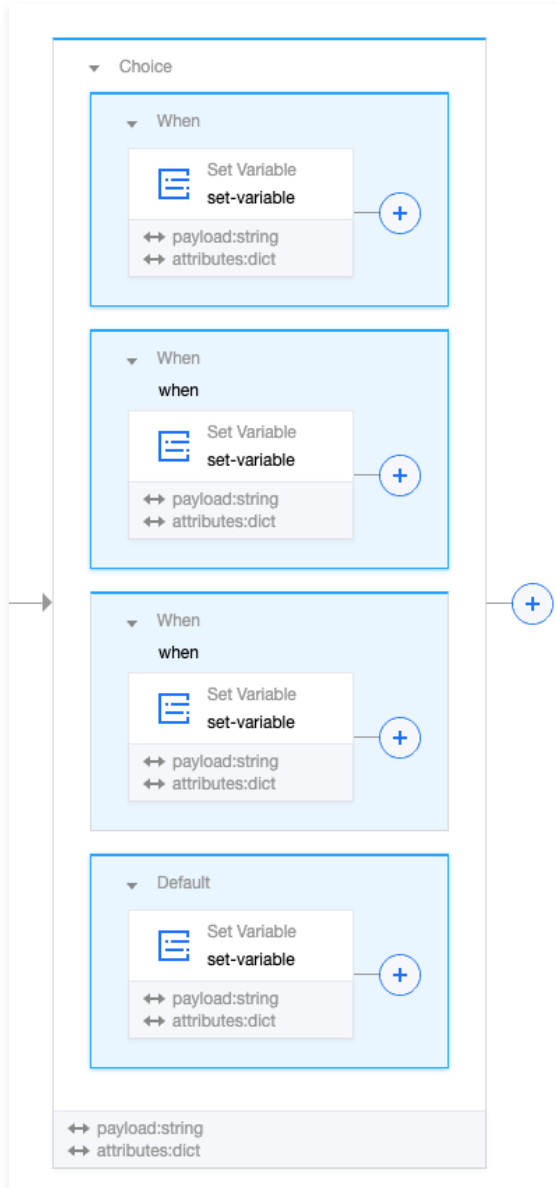
2. 在 Choice 中添加 When 节点，每个 When 节点都是一个分支条件。



3. 在 When 节点中填入判断条件，第一个 When 节点输入 `msg.vars.get('score') >= 90`，第二个 When 节点输入 `msg.vars.get('score') >= 80`，第三个 When 节点输入 `msg.vars.get('score') >= 60`，Default 节点中不用输入条件。



4. 在每个 When 节点中，添加 Set Variable 节点，保存 level 值。



Set Variable ?

...

[-]

配置

备注

基本配置

▲

变量名 ·

string ▾

grade

变量值 ·

string ▾

A

f(x)

Continue

最近更新时间：2023-06-08 14:47:55

简介

Continue 组件类似于 Break 组件，搭配 For Each 和 While 组件使用。Continue 的作用是跳出当前循环，执行下一次循环。

操作配置

参数配置

无

输出

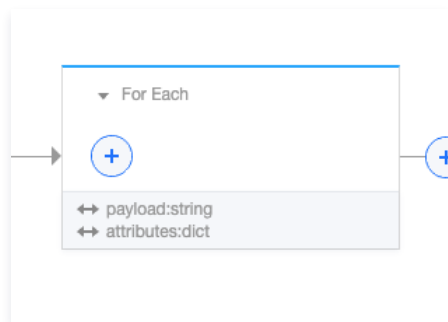
组件输出的 message 信息如下：

message 属性	值
payload	在 For Each 组件中，Continue 跳出当前循环后，payload 的值为下一次循环遍历的数据。在 While 组件中，Continue 跳出循环后，payload 继承自 Continue 的上一个组件输出的 payload。
error	无。
attribute	在 For Each 组件中使用时，Continue 跳出循环后，attribute 继承自 For Each 的上一个组件输出的 attribute；在 While 组件中使用时，Continue 跳出循环后，attribute 继承自 While 的上一个组件输出的 attribute。
variable	在 For Each 中使用时，Continue 跳出循环后，variable 中的变量继承自 Continue 的上一个组件输出的 variable 变量；在 While 组件中使用时，variable 中的变量继承自 Continue 的上一个组件输出的 variable 变量。

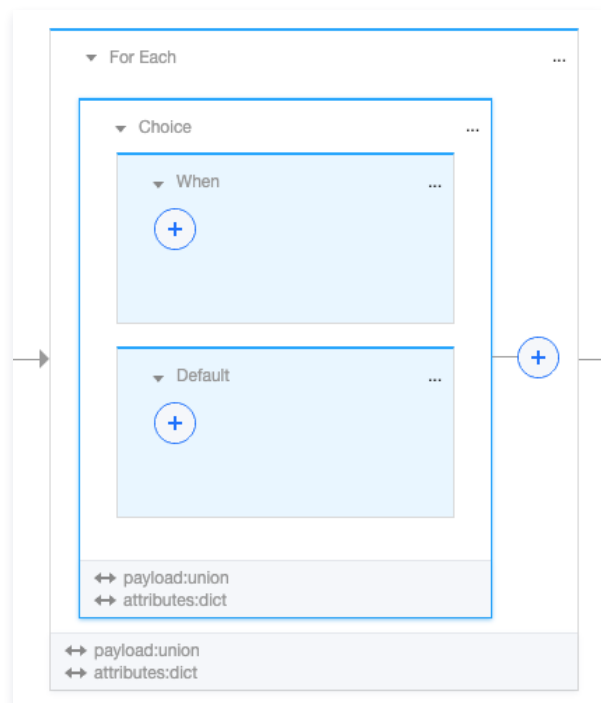
案例

在本案例中，我们使用 For Each 对列表中的奇数求和，遇到偶数，则使用 Continue 跳出当次循环。

1. 添加 For Each 组件，填入要遍历的集合[1,2,3,4,5]。



2. 添加 Choice 组件，在 When 节点中对数据进行筛选。



Choice ?

配置

备注

配置 / 编辑

Dataway表达式 ?

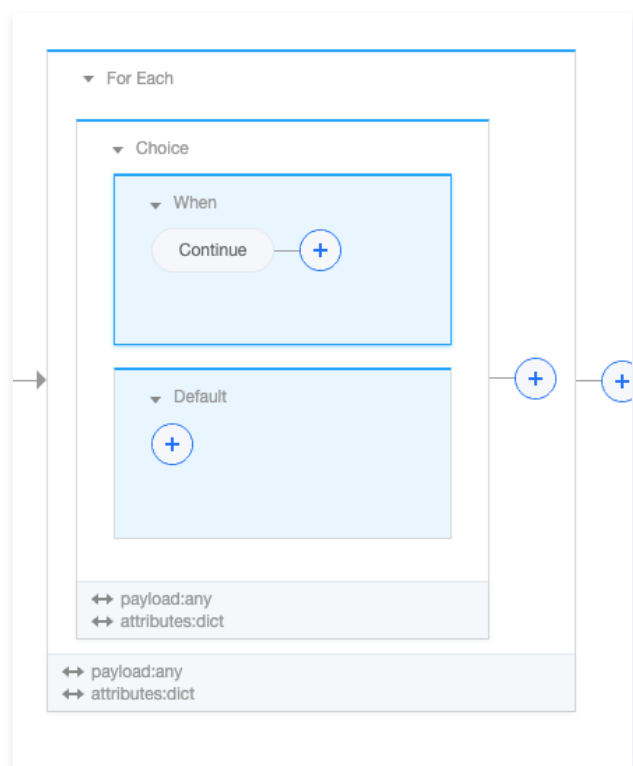
1

def dw_process(msg):

2

return msg.payload %2 == 0

3. 在 When 节点中加入 Continue 节点。



Crypto

最近更新时间：2024-12-11 16:27:42

简介

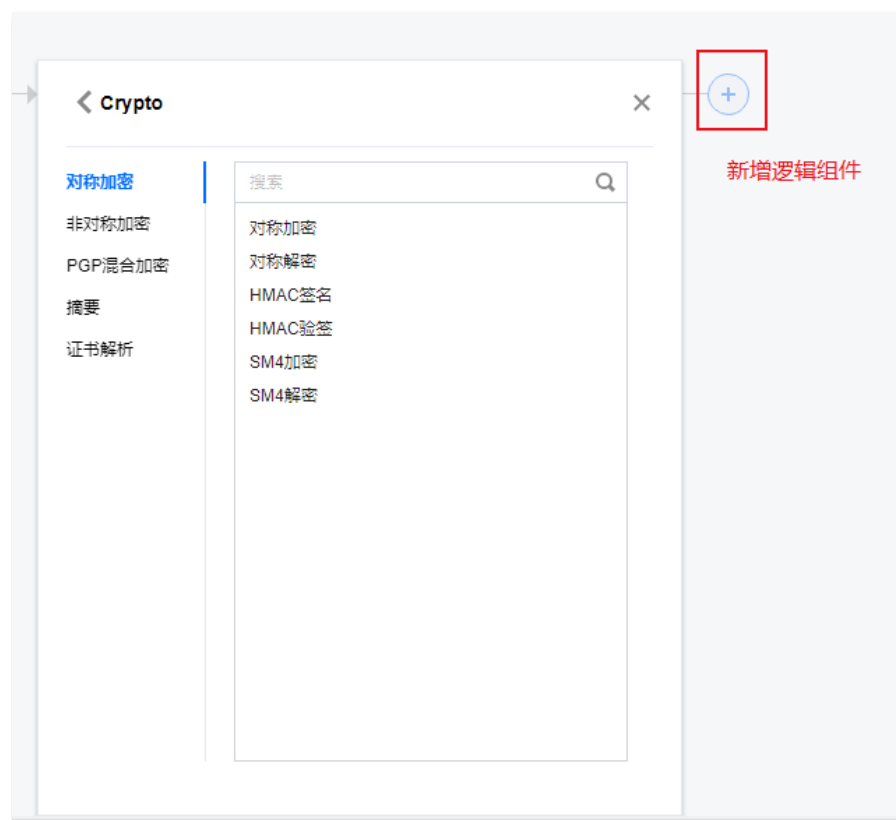
Crypto 连接器提供常用的对称、非对称和 PGP 加解密功能。

操作说明

连接器配置说明

步骤1：创建连接器

单击 + 弹出组件筛选框，选择 **Crypto** 组件。根据业务场景选择不同的操作类型。



步骤2：新建连接配置

1. 单击新建，创建 Crypto 连接器配置。



Crypto 对称加密

当前暂未绑定任何配置，请选择或者新建连接配置

新建 | 绑定

配置 备注

配置页面根据选择的操作不同参数不同
参考文档进行详细配置

通用

加密算法: AES

加密模式: CBC

随机向量: string 请输入字符串 f(x)

明文: 请输入表达式

内容填充模式: PKCS5_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性: payload

类型选择: 空 +

2. 设置配置名称，用户自定义即可，并单击下一步。



新建连接器配置

1 选择连接器 > 2 配置连接器

连接器类型: Crypto

配置名称: 请输入配置名称

下一步 取消

3. 配置加密方式及密钥配置。

新建连接器配置

✓ 选择连接器

>

2 配置连接器

通用

加密方式

对称加密

对称加密

非对称加密

PGP混合加密

密钥配置

密钥填充模式

NO_PADDING

密钥

f(x)

上一步

保存

对应不同加密方式
密钥配置参数不同

参数	数据类型	描述	是否必填	默认值
加密方式	enum	对称加密、非对称加密、PGP 混合加密	是	-

不同加密方式对应的配置如下：

对称加密配置

参数	数据类型	描述	是否必填	默认值
密钥填充模式	enum	PBKDF2、NO_PADDING、ZERO_PADDING	是	PBKDF2

PBKDF2 模式

参数	数据类型	描述	是否必填	默认值
PBKDF2 哈希算法	enum	MD5、SHA1、SHA224、SHA256、SHA384、SHA512	是	SHA256
PBKDF2 盐值	string	PBKDF2 盐值	是	-
PBKDF2 迭代次数	int	PBKDF2 迭代次数	是	0
口令	string	密码	是	-

NO_PADDING 或 ZERO_PADDING 模式

参数	数据类型	描述	是否必填	默认值
密钥	string	加密密钥	是	-

⚠ 注意：

密钥长度限制与组件配置-通用中选择的加密算法有关，AES/16字节，DES/8字节，3DES/24字节。

PGP 加密配置

参数	数据类型	描述	是否必填	默认值
PGP 私钥证书	string	PGP 私钥文件	是	-
PGP 公钥证书	string	PGP 公钥证书	是	-
PGP 钥匙串	map	指纹和密码对信息	是	-

非对称加密配置

参数	数据类型	描述	是否必填	默认值
私钥证书	string	私钥文件	是	-
公钥证书	string	公钥证书	是	-

组件配置说明

Crypto 组件目前支持对称加密、对称解密、非对称加密、非对称解密、PGP 加密、PGP 解密、PGP 签名、PGP 验签等操作，连接器配置新建完毕后进行组件的通用配置。

对称加密

参数配置

参数	数据类型	描述	是否必填	默认值
加密算法	enum	AES、DES、3DES	是	AES
加密模式	enum	CBC、ECB	是	CBC
随机向量	string	CBC加密模式必填，长度跟密钥长度一致	CBC加密模式必填	-
明文	entity	待加密内容	是	-
内容填充模式	enum	PKCS5_PADDING、ZERO_PADDING、PKCS7_PADDING、NO_PADDING	是	PKCS5_PADDING

CBC 加密模式

选择 CBC 加密模式时，随机向量必填，长度跟 [密钥长度](#) 一致，再填写明文和填充模式即可。

配置

备注

通用

加密算法

AES

加密模式

CBC

随机向量

string

1234567812345678

f(x)

明文

def dw_process(msg): return Entity.from_bytes

内容填充模式

PKCS5_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

ECB 加密模式

当选择 ECB 加密模式时，只需填写明文和填充模式，无其他配置项。

配置

备注

通用

加密算法

AES

加密模式

ECB

明文

def dw_process(msg): return Entity.from_bytes

内容填充模式

PKCS5_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回加密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息

variable	继承上个组件的 variable 信息
----------	---------------------

对称加密案例

1. 添加 Crypto 连接器组件，选择对称加密操作。



2. 在配置中，填写相关参数。例如：待加密内容使用 Dataway 表达式输入：

```
def dw_process(msg) :  
    return Entity.from_bytes('test', mime_type='text/plain', encoding='utf-8')
```

配置界面如下：



3. 执行成功后，message payload 中包含了加密后的二进制内容：

输出消息 payload

[

247,

246,

154,

244,

35,

71,

20,

227,

100,

108,

202,

50,

75

]

下载

对称解密

参数配置

参数	数据类型	描述	是否必填	默认值
加密算法	enum	AES、DES、3DES	是	AES
加密模式	enum	CBC、ECB	是	CBC
随机向量	string	CBC加密模式必填，长度跟密钥长度一致	CBC加密模式必填	-
明文	entity	待解密内容	是	-
内容填充模式	enum	PKCS5_PADDING、ZERO_PADDING、PKCS7_PADDING、NO_PADDING	是	PKCS5_PADDING

CBC 加密模式

选择 CBC 加密模式时，随机向量必填，长度跟 密钥长度 一致，再填写明文和填充模式即可。

配置

备注

通用

加密算法

AES

加密模式

CBC

随机向量

string

1234567812345678

f(x)

明文

def dw_process(msg): return Entity.from_bytes

内容填充模式

PKCS5_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

ECB 加密模式

当选择 ECB 加密模式时，只需填写明文和填充模式，无其他配置项。

配置

备注

通用

加密算法

AES

加密模式

ECB

明文

def dw_process(msg): return Entity.from_bytes

内容填充模式

PKCS5_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回解密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

对称解密案例

1. 添加 Crypto 连接器组件，选择对称解密操作。

< Crypto

×

对称加密

非对称加密

PGP混合加密

摘要

证书解析

搜索

Q

对称加密

对称解密

HMAC签名

HMAC验签

SM4加密

SM4解密

2. 在配置中，填写相关参数。例如：待解密内容使用 Dataway 表达式输入：

```
def dw_process(msg) :  
    return Entity.from_bytes(msg.payload, mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

配置

调试

备注

通用

加密算法

AES

加密模式

CBC

明文

def dw_process(msg): return Entity.from_bytes

内容填充模式

PKCS5_PADDING

3. 执行成功后，message payload 中包含解密后的二进制内容：

输出消息 payload

```
[
  116,
  101,
  115,
  116
]
```

下载

非对称加密

参数配置

参数	数据类型	描述	是否必填	默认值
加密算法	enum	RSA	是	RSA
明文	entity	待加密内容	是	-
填充模式	enum	RSA_PKCS1_PADDING、RSA_PKCS1_OAEP_PADDING	是	RSA_PKCS1_PADDING

RSA_PKCS1_OAEP_PADDING 模式

内容填充模式选择“RSA_PKCS1_OAEP_PADDING”时，配置如下：

参数	数据类型	描述	是否必填	默认值
OAEP 哈希算法	enum	MD5/SHA1、SHA224、SHA256、SHA384、SHA512	是	SHA256

Crypto 非对称加密

当前使用的是 非对称加密 连接配置

编辑 | 切换

配置备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_OAEP_PADDING

OAEP 哈希算法

SHA256

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

RSA_PKCS1_PADDING 模式
内容填充模式选择“RSA_PKCS1_PADDING”时，无其他配置。

配置备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回加密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息

variable	继承上个组件的 variable 信息
----------	---------------------

非对称加密案例

1. 添加 Crypto 连接器组件，选择非对称加密操作。



2. 在配置中，填写相关参数。例如：待加密内容使用 Dataway 表达式输入：

```
def dw_process(msg):  
    return Entity.from_bytes('test', mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

配置

备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

3. 执行成功后，message payload 中包含了加密后的二进制内容：



非对称解密

参数配置

参数	数据类型	描述	是否必填	默认值
加密算法	enum	RSA	是	RSA
明文	entity	待解密内容	是	-
填充模式	enum	RSA_PKCS1_PADDING、RSA_PKCS1_OAEP_PADDING	是	RSA_PKCS1_PADDING

RSA_PKCS1_OAEP_PADDING 模式

参数	数据类型	描述	是否必填	默认值
OAEP 哈希算法	enum	MD5/SHA1、SHA224、SHA256、SHA384、SHA512	是	SHA256

配置

备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_OAEP_PADDING

OAEP 哈希算法

SHA256

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

RSA_PKCS1_PADDING 模式

内容填充模式选择 “RSA_PKCS1_PADDING” 时, 无其他配置。

Crypto ? 非对称加密

当前使用的是 非对称加密 连接配置

编辑 | 切换

配置

备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回解密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

非对称解密案例

1. 添加 Crypto 连接器组件，选择非对称解密操作。

< Crypto

×

对称加密

非对称加密

PGP混合加密

摘要

证书解析

搜索

非对称加密

非对称解密

非对称签名

非对称验签

SM2加密

SM2解密

SM2签名

SM2验签

2. 在配置中，填写相关参数。例如：待解密内容使用 Dataway 表达式输入：

```
def dw_process(msg):
    return Entity.from_bytes(msg.payload, mime_type='text/plain', encoding='utf-8')
```

版权所有：腾讯云计算（北京）有限责任公司

第69 共732页

配置界面如下：

Crypto 非对称加密

当前使用的是 非对称加密 连接配置

配置

备注

通用

加密算法

RSA

明文

def dw_process(msg): return Entity.from_bytes

填充模式

RSA_PKCS1_PADDING

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

3. 执行成功后，message payload中包含了解密后的二进制内容：

输出消息 payload

[

116,

101,

115,

116

]

下载

PGP 加密

参数配置

参数	数据类型	描述	是否必填	默认值
钥匙串 ID	string	钥匙串 ID	是	-
加密算法	enum	CAST5、3DES、AES_128、AES_192、AES_256	是	AES_128
内容	entity	待加密内容	是	-

密钥串ID

string

6891053952471327761

f(x)

加密算法

AES_256

内容

def dw_process(msg): return Entity.from_bytes

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回加密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

PGP 加密案例

1. 添加 Crypto 连接器组件，选择 PGP 加密操作。

< Crypto

对称加密

非对称加密

PGP混合加密

搜索

PGP加密

PGP解密

PGP签名

PGP验签

2. 在配置中，填写相关参数。例如：待加密内容使用 Dataway 表达式输入：

```
def dw_process(msg):  
    return Entity.from_bytes("123456", mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

密钥串ID

string

6891053952471327761

f(x)

加密算法

AES_256

内容

def dw_process(msg): return Entity.from_bytes

3. 执行成功后，message payload 中包含了加密后的二进制内容：

输出消息

payload

69,

45,

45,

45,

45,

45,

10,

10,

119,

89,

119,

68,

110,

74

下载

PGP 解密

参数配置

参数	数据类型	描述	是否必填	默认值
密钥串 ID	string	密钥串 ID	是	-
内容	entity	待解密内容	是	-

通用

密钥串ID

string

6891053952471327761

f(x)

内容

def dw_process(msg): return Entity.from_bytes

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回解密后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

PGP 解密案例

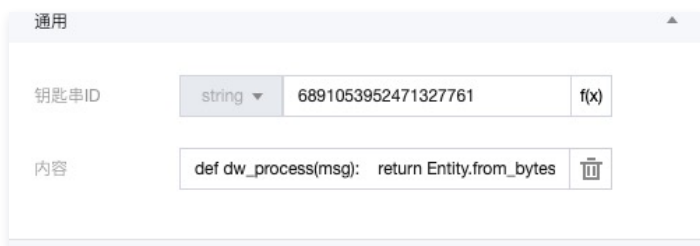
1. 添加 Crypto 连接器组件，选择 PGP 解密操作。



2. 在配置中，填写相关参数。例如：待解密内容使用 Dataway 表达式输入：

```
def dw_process(msg):
    return Entity.from_bytes(msg.payload, mime_type='text/plain', encoding='utf-8')
```

配置界面如下：



3. 执行成功后，message payload 中包含了加密后的二进制内容：



PGP 签名

参数配置

参数	数据类型	描述	是否必填	默认值
钥匙串 ID	string	钥匙串 ID	是	-
签名算法	enum	MD5、SHA1、SHA224、SHA256、SHA384、SHA512	是	AES_256
内容	entity	待签名内容	是	-

通用

钥匙串ID

string ▼ 6891053952471327761 f(x)

签名算法

SHA256 ▼

内容

def dw_process(msg): return Entity.from_bytes

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回签名后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

PGP 签名案例

1. 添加 Crypto 连接器组件，选择 PGP 签名操作。

< Crypto

对称加密

非对称加密

PGP混合加密

搜索

PGP加密

PGP解密

PGP签名

PGP验签

2. 在配置中，填写相关参数。例如：待签名内容使用 Dataway 表达式输入：


```
def dw_process(msg):
    return Entity.from_bytes("123456", mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

通用

钥匙串ID

string

6891053952471327761

f(x)

签名算法

SHA256

内容

def dw_process(msg): return Entity.from_bytes

🗑️

3. 执行成功后，message payload 中包含了签名后的二进制内容：

输出消息 payload

45,

45,

45,

45,

45,

66,

69,

71,

73,

78,

32,

80,

74

下载

PGP 验签

参数配置

参数	数据类型	描述	是否必填	默认值
钥匙串 ID	string	钥匙串 ID	是	—
目标内容	entity	待比较内容	是	—
期望内容	entity	签名内容	是	—

通用

钥匙串ID

string

请输入字符串

f(x)

目标内容

def dw_process(msg): return Entity.from_bytes

🗑️

期望内容

def dw_process(msg): return Entity.from_bytes

🗑️

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 为 binary 类型，返回验证签名后的结果；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

PGP 验签案例

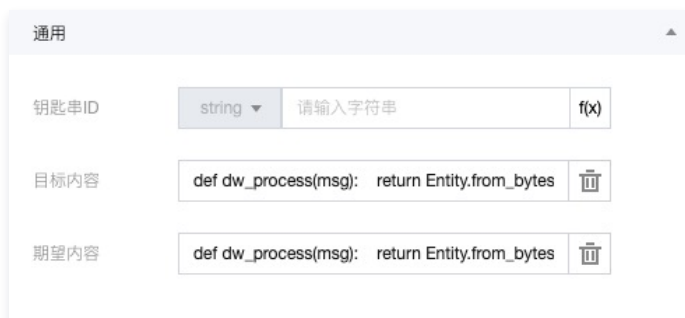
1. 添加 Crypto 连接器组件，选择 PGP 验签操作。



2. 在配置中，填写相关参数。例如：期望内容、目标内容使用 Dataway 表达式输入：

```
def dw_process(msg):
    return Entity.from_bytes(msg.payload, mime_type='text/plain', encoding='utf-8')
def dw_process(msg):
    return Entity.from_bytes("123456", mime_type='text/plain', encoding='utf-8')
```

配置界面如下：



3. 执行成功后，message payload 中包含验证签名是否正确结果：



Flow Reference

最近更新时间：2024-08-21 16:44:01

简介

Flow Reference 组件用来引用当前项目下集成应用中的其他集成流。用户可以通过 Flow Reference 组件引用同一项目下不同应用的集成流或当前应用中的其他集成流，减少重复配置的操作。

与 Async 不同，Flow Reference 是一个同步动作，当引用的集成流执行完成后，才会继续执行下一个动作，并且 Flow Reference 中的子流执行完成后，message 会传递到主流中，下一个节点基于该子流的 message 继续执行。当 Flow Reference 引用的子流包含 trigger 节点时，如果该子流的执行由 Flow Reference 触发，则该子流的 trigger 节点不会执行，即该子流从第二个节点开始执行。

操作说明

参数配置

参数	数据类型	描述	是否必填	默认值
集成流	String	集成流名称	是	无

配置界面



输入到子流中的 message

message 属性	值
payload	继承 Flow Reference 上一个组件的 payload。
error	空。
attribute	继承 Flow Reference 上一个组件的 attribute 信息。
variable	继承 Flow Reference 上一个组件的 variable 信息。

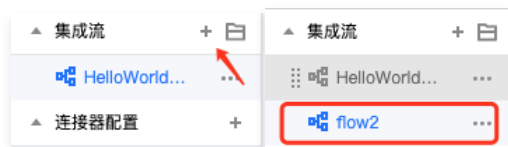
输出

Flow Reference 组件的执行结果，是子流的最后一个组件的输出。组件输出的 message 信息如下：

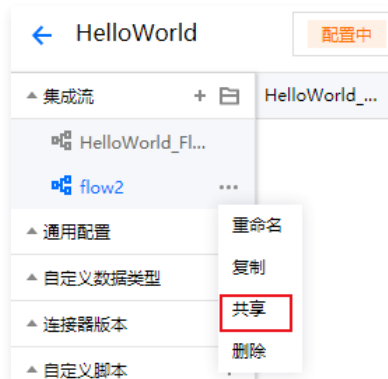
message 属性	值
payload	继承 Flow Reference 中子流输出的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承 Flow Reference 中子流输出的 attribute。
variable	继承 Flow Reference 中子流输出的 variable。

案例

1. 新建一条集成流，名称为 flow2。



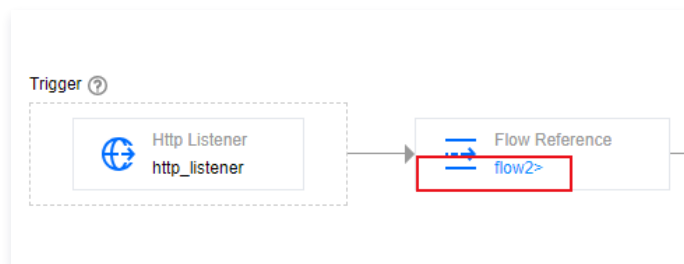
2. 对需要被引用的流，设置共享集成流的操作。



3. 添加 Flow Reference 组件，在下拉框中选择流 flow2，则对当前集成流设置引用 flow2 集成流完成。



完成后界面如下：



For Each

最近更新时间：2023-06-08 14:47:55

简介

For Each 是循环控制结构，类似于编程语言中的 for/foreach，在 For Each 中可以配置子流，对指定的数据集进行循环处理，但是无法指定循环次数。使用该组件时，数据集填写待遍历的数据，计数器中填入一个变量名称，默认为 counter，该变量存储当前的迭代次数。根信息中同样填入一个变量名称，默认为 rootMessage，该变量中保存主流的 message 信息，For Each 中配置的子流可使用 msg.vars.get('rootMessage') 访问主流中的 message。

操作说明

参数配置

参数	数据类型	描述	是否必填	默认值
数据集	string、list、dict、int	待遍历的数据： - 当类型为 string 时，遍历字符串的每个字符。 - 当类型为 list 时，遍历 list 的每个元素。 - 当类型为 dict 时，遍历 dict 中的 value。 - 当数据集类型为 int 时，例如3，实际遍历的数据集为[0,1,2]。	是	无
计数器	string	计数器是一个变量，该变量存储当前的迭代次数，从0开始，填入变量名称，msg.vars.get('#计数器变量#') 即可使用；例如：当计数器变量使用默认值 counter 时，第1次循环，msg.vars.get('counter') 值为0，第2次循环，msg.vars.get('counter') 值为1。	否	counter
根信息	string	根信息同样是一个变量，填入变量名称，根信息中保存主流的 message 信息，msg.vars.get('#根信息名称#').payload 即可访问主流的 payload 数据。当使用默认值 rootMessage 时，使用 msg.vars.get('rootMessage').payload 即可在 For Each 的子流中访问主流的 payload 数据。	否	rootMessage

配置界面

For Each ?

配置 输入 输出 备注

...

□

基本配置

数据集

string

输入字符串

组件遍历循环的数据集对象

计数器

请输入计数器

循环遍历的计数器

根信息

请输入根信息

用于指定循环遍历的父级流消息对象

输入到子流中的 message

message 属性	值
payload	数据集集中的元素，例如：待迭代的数据集为[1,2,3]，第一次循环，子流中 payload 的数据为1，第二次循环为2；待迭代的数据集为 dict 类型，{"key":"key1", "value":"value1"}，第一次循环，子流中的 payload 为 key1，第二次为 value1。
error	空。
attribute	空。
variable	继承主流中的 variable 数据，同时新增两个变量，一个是计数器，一个是根信息，若用户使用默认值，可使用表达式 msg.vars.get('counter') 和 msg.vars.get('rootMessage') 访问。若 For Each 中使用了 Set Variable，则在子流执行过程中，新增的变量也会添加到 variable 中。

输出

For Each 组件输出的 message 中，保留了 For Each 上一个组件的 payload 和 attributes 信息，只有 For Each 中子流使用的 variable 变量会传递出来。组件输出的 message 信息如下：

message 属性	值
payload	继承 For Each 上一个组件输出的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上 For Each 上一个组件输出的 attribute 信息。
variable	For Each 上一个组件输出的 variable 变量加上 For Each 子流中声明的 variable 变量。

案例

使用 For Each 组件遍历列表，为列表中的元素添加序号及前缀，列表元素及前缀从 HTTP 请求的参数中获得，操作步骤如下：

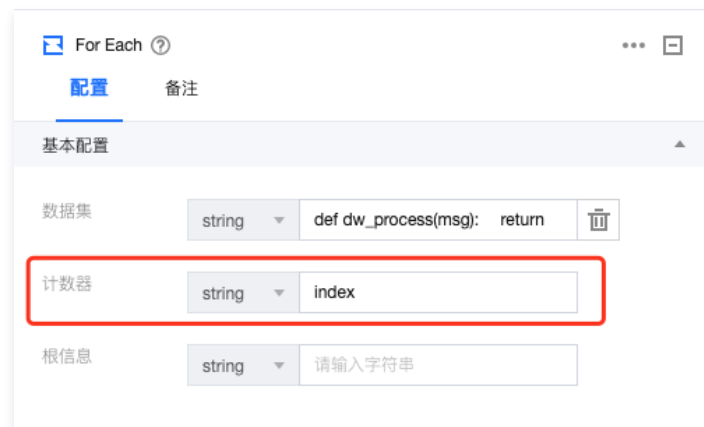
1. 添加 For Each 组件，数据集从 HTTP 请求的参数中获得，计数器设置为“index”，根信息使用默认变量“rootMessage”。



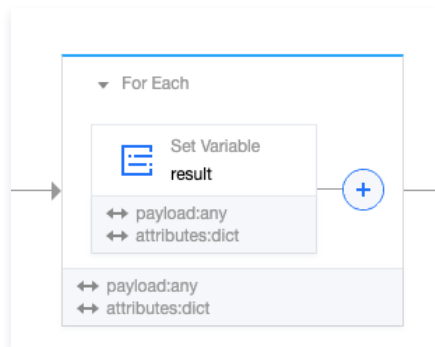
2. 配置数据集。



3. 设置计数器。



4. 在 For Each 组件中加入 Set Variable 组件，用于处理业务逻辑。



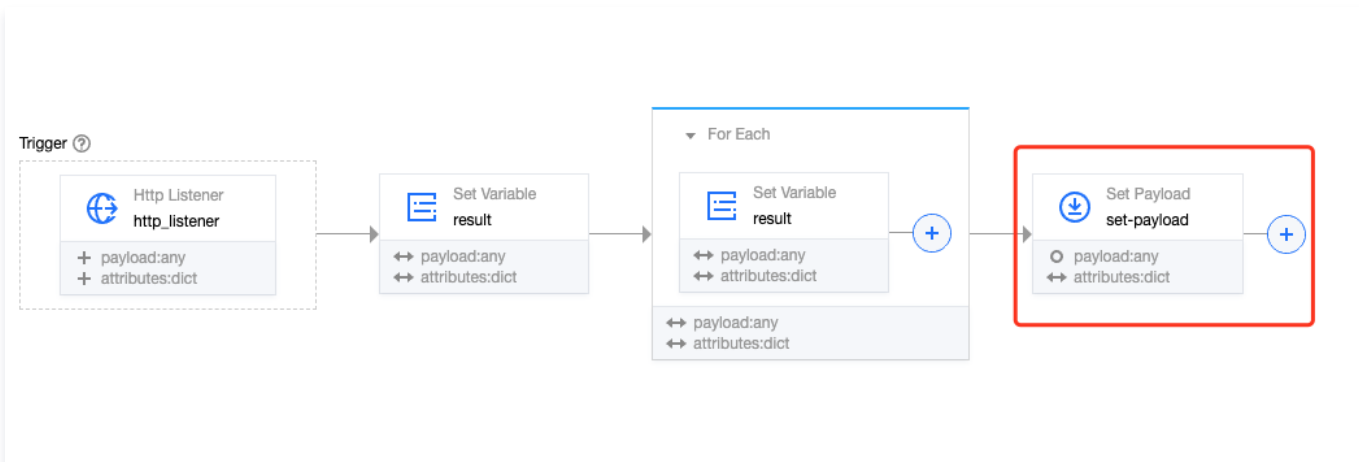
如下图所示，第二行是从 msg.vars 中获取当前的索引值，第三行是从 HTTP 请求中获取前缀，第四行将拼接完成的字符串赋值到 result 变量中。

Dataway表达式 ?

```

1 def dw_process(msg):
2     index = str(msg.vars.get('index'))
3     prefix = msg.vars.get('rootMessage').attrs.get('queryParams').get('prefix')
4     msg.vars.get('result')[index] = prefix + ':' + msg.payload
5     return msg.vars.get('result')
    
```

5. 在 For Each 组件后加入 Set Payload 组件，将结果输出。



Set Payload ?

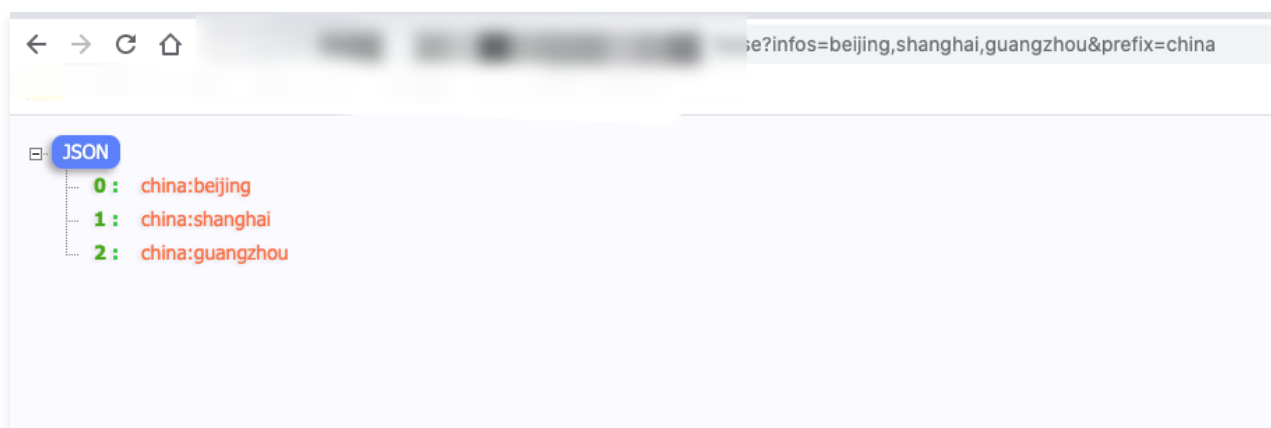
配置 备注

配置 / 编辑

Dataway表达式 ?

```

1 def dw_process(msg):
2     return msg.vars.get['result']
    
```

6. 输出效果。

Filter

最近更新时间：2024-10-21 17:35:22

简介

RecordSet Filter 组件用于操作 RecordSet，可以对输入的 RecordSet 进行过滤，并生成一个新的 RecordSet。

操作配置

参数配置

配置数据源

参数	数据类型	描述	是否必填	默认值
输入数据集	RecordSet	RecordSet Filter 组件的数据源	否	msg.payload

基本配置

输入数据集



配置输出字段

用于指定新生成的 RecordSet 中要包含的字段。默认将数据源的字段罗列出来，用户只需要勾选希望输出的字段即可。用户也可以给输出的 RecordSet 注入额外的字段，且字段的默认值将为空。

字段选择



字段	字段格式	操作
☒ name	string	
☒ age	int	
☒ address	string	
☐ score	flot	

[+新增字段](#)

配置过滤条件

用于配置字段需要满足的条件，支持图形化配置（默认）和表达式两种方式。

图形化配置（默认）

如果使用图形化配置（默认）方式，用户可以给数据源中的字段配置1到多个原子过滤条件，原子过滤条件间默认是逻辑“与”的关系。

- 每个原子过滤条件都由三部分组成：字段名、操作、目标值。
- 字段名：数据源中的字段名。
- 操作：支持 ==, !=, >=, <=, >, <, StartWith, EndWith, Contain, In, Not In, IsNull, IsNotNull ，详见下表。

操作	说明
==	判断相等
!=	判断不相等

>=	比较大小
<=	比较大小空
>	比较大小空
<	比较大小空
StartWith	判断字符串是否以另一个字符串开头
EndWith	判断字符串是否以另一个字符串结尾
Contain	判断字符串包含
In	判断是否在列表中
NotIn	判断是否不在列表中
IsNull	判断是否为空
IsNotNull	判断是否非空

- 目标值：支持手动填写和列表选择两种方式。手动填写时，可以将目标值设置常量或输入表达式；列表选择时可以将目标值设置为数据源中其他字段。

设置目标值

×

设置方式 ☒ 手动填写 ☐ 列表选择

目标值

string ▼

eedwinzeng

f(x)

确定

取消

设置目标值

×

设置方式 ☐ 手动填写 ☒ 列表选择

目标值

请选择 ▼

name

age

address

score

adult

确定

取消

- 原子过滤条件间默认是逻辑“与”的关系。如果用户需要配置更复杂的逻辑，可以使用编排逻辑功能。
- 每个原子过滤条件有一个唯一的序号，用户在编排逻辑输入框内，用序号对原子过滤条件进行逻辑编排。
- 支持逻辑与（And）、或（Or）、非（Not）的组合，且忽略大小写。

- 用户也可以用括号来控制优先级。

序号	字段名	操作	目标值	操作
1	name	==	eedwinzeng	删除
2	age	<	score	删除
3	address	Contain	湖北	删除
4	score	>	60	删除
5	adult	!=	false	删除

+添加变量

编排逻辑

(1or2)and not3 and4 and5

点击校验

- 示例

```
(1 OR 2) AND NOT 3
(1 Or 2) AnD nOT 3
(1 AND 2) AND NOT 3
1 or 2 and (not 3)
not 1 or 2 or 3
not (1 or 2 or 3)
(not 1 or 2) and 3
1 and not( 2 or 3)
1or2and3or4
```

表达式

当图像化配置过滤条件无法满足业务需求时，可以使用表达式输入进行补充。用户需要勾选表达式，并编辑一个返回 bool 类型的 Dataway 表达式。

字段选择

搜索字段名称

字段	字段格式	操作
☒ name	string	删除
☒ age	int	删除
☒ address	string	删除
☐ score	flot	删除

+新增字段

字段过滤

☒ 表达式
 过滤条件

string
 def dw_process(msg): f1 = msg.paylo

表达式里面的内容示例如下：

Dataway表达式 [?](#)

```
1 def dw_process(msg):
2     f1 = msg.payload["name"]=="xxxx"
3     f2 = msg.payload["age"] < msg.payload["score"]
4     f3 = "湖北" in msg.payload["address"]
5     f4 = msg.payload["score"] > 60
6     f5 = msg.payload["adult"] != False
7     return f1 and f2 and f3 and f4 and f5
```

输入 message

message 属性	值
payload	默认是 payload，如果用户有配置，则采用用户配置的值
error	空
attribute	空
variable	继承上一个组件输出 message 中的 variable 数据，在 RecordSet Filter 组件中可以操作 Variables，且不会影响输出 message 中的数据

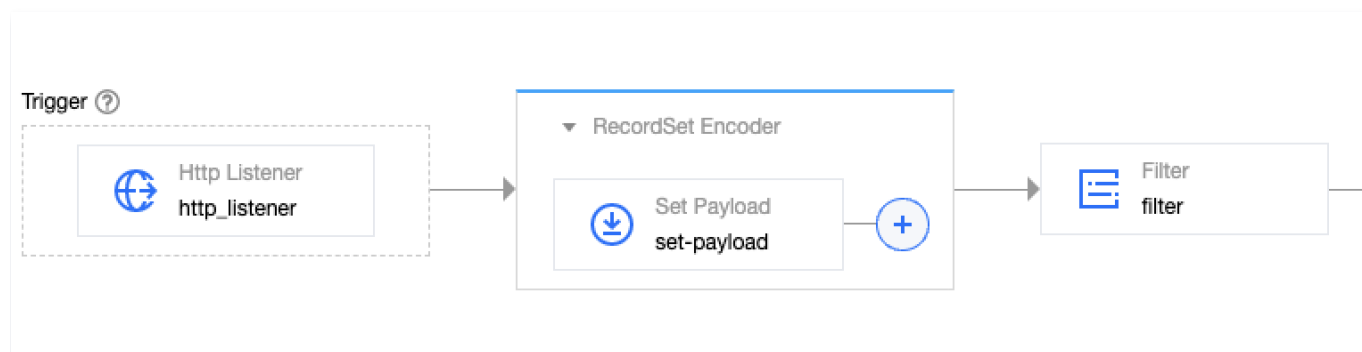
输出

组件输出的 message 信息如下：

message 属性	值
payload	输出的 RecordSet，其包含的字段为用户配置的输出字段
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	类型为 dict，继承上一个组件的 attribute
variable	继承上一个组件的 variable

案例

针对一个学生表按指定条件进行过滤，示例图如下。



1. 用 RecordSet Encoder 组件创建一个学生表，包含 name, age, address, score, adult 字段，并在 RecordSet Encoder 组件里面添加 SetPayload 组件，在学生表中使用表达式添加数据。

RecordSet ... ?

配置 备注

基本配置

输出Schema

序号	字段类型	字段名	显示名称	操作
1	string	name	name	删除
2	int	age	age	删除
3	string	address	address	删除
4	float	score	score	删除
5	bool	adult	adult	删除

+添加变量

Dataway表达式 ?

```

1 def dw_process(msg):
2     return [
3         ['张三', 10, '陕西省西安市', 99.5, False],
4         ['张三', 20, '湖北省荆州市', 61.2, True],
5         ['张三', 25, '湖北省洪湖市', 71.1, True],
6         ['张三', 30, '湖北省赤壁市', 30.33, True],
7         ['张三', 40, '湖北省襄阳市', 80.33, False ],
8         ['张三', 80, '湖北省襄阳市', 40.33, True ],
9         ['张三', 90, '湖北省襄阳市', 30.33, False ],
10        ['李四', 70, '北京省长江市', 10.33, True],
11        ['王五', 80, '广东省深圳市', 70.33, True],
12        ['王五', 30, '湖南省长沙市', 73.33, True],
13        ['王五', 40, '湖南省长沙市', 23.33, False],
14    ]
    
```

2. 用 Filter 组件进行过滤，生成一个新的学生表，仅包含 name, age, address 3个字段，并配置5个原子过滤条件，最后对原子过滤条件进行逻辑编排 (1or2)and not3 and4 and5，其中每个数字都代表对应序号的原子过滤条件。

Filter ?

...

☐

配置

备注

基本配置

▲

输入数据

请输入表达式

☒

字段选择

▲

搜索字段名称

Q

☐

 字段

字段格式

操作

☑

 name

string

☒

☑

 age

int

☒

☑

 address

string

☒

☐

 score

flot

☒

+新增字段

字段过滤

▲

☐

 表达式

序号

字段名

操作

目标值

操作

1

name

==

张三

删除

2

age

<

score

删除

3

address

Contain

湖北

删除

4

score

>

60

删除

5

adult

!=

false

删除

+添加变量

编排逻辑

(1or2)and not3 and4 and5

点击校验 ↻

高级配置

▲

输出持久化

☑

3. 在调试模式下进行单元测试，并查看结果。输出的结果将只包含一个学生的数据： ["王五", 30, "湖南省长沙市"]

单元测试

1 请选择所要测试的一个或多个组件

2 请模拟单元测试输入数据

Http Listener
http_listener

Encoder

Set Payload
set-payload

Filter
filter

取消

下一步

版权所有：腾讯云计算（北京）有限责任公司

第90 共732页

单元测试

1 请选择所要测试的一个或多个组件

2 请模拟单元测试输入数据

☐ 使用首个组件的历史输入数据作为单元测试的输入数据

payload 值

string

请输入字符串

attribute 值

变量名

变量值

操作

+添加变量

variable 值

变量名

变量值

操作

+添加变量

上一步

开始测试

取消

测试中 20210623_9.0

Trigger ①

HTTP Listener

http_listener

RecordSet Encoder

Set Payload

set-payload

Filter

filter

Filter ①

配置

调试

备注

运行消息

历史消息

2021-06-24 21:40:58.47

输入消息 payload

```
{
  "id": 41,
  "partition": 1,
  "records": [
    {
      "张三",
      10,
      "陕西省西安市",
      99.5,
      false
    }
  ],
  "张三",
  ...
}
```

下载

输出消息 payload

```
{
  "id": 43,
  "partition": 1,
  "records": [
    {
      "王五",
      30,
      "湖南省长沙市"
    }
  ],
  "schema": {
    "name(string)",
    "age(int)",
    ...
  }
}
```

下载

Logger

最近更新时间：2023-06-08 14:47:55

简介

Logger 组件用来在控制台输出日志，目前支持 DEBUG、INFO、WARN、ERROR 四种日志级别。

操作说明

参数配置

参数	数据类型	描述	是否必填	默认值
日志级别	string	日志级别，目前支持 DEBUG、INFO、WARN、ERROR，在运行监控中查看日志时，可以根据级别筛选日志。	是	INFO
日志类别	string	日志分类，由用户自定义。	否	无
日志内容	any	日志内容可以是 dataway 支持的任意类型，在输出时，系统会自动转为 string。	是	无

配置界面

Logger ?

配置 输入 输出 备注

基本配置

日志级别 *

INFO

日志的级别

日志类别

日志的类别

日志内容 *

f(x)

单击输入DataWay表达式

输出

Logger 组件只是输出内容到控制台，并不会改变 message 中的内容，输出内容中包括日志级别、日志类别、日志内容。组件输出的 message 信息如下：

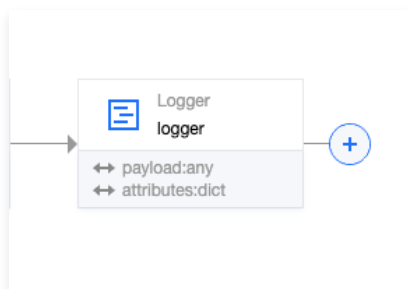
message 属性	值
payload	继承上个组件的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	继承上个组件的 variable 信息。

应用名	集成流名	日志级别	时间	日志数据
ipaaS白皮书	NewFlow	WARN	2021/3/25 上午11:08:22	{"category":"ipaaS","componentId":"4","date":"2021-03-25 11:08:22.058","docId":"23b7b04f-c446-4341-be04-96d2ddf31686","instanceId":"94897","messageSeqId":"a5a...

列表	JSON
category	ipaaS
componentId	4
date	2021-03-25 11:08:22.058
messageSeqId	a5ae4ccc-072b-4231-9ea0-65e1a4eda38d
msg	null
project	ipaaS

案例

1. 添加 Logger 组件。



2. 在日志级别的下拉列表中，选择 INFO，日志内容中填入待打印的数据。

Logger ?

...

[-]

配置

备注

基本配置

▲

日志级别 *

INFO ▼

日志类别

请输入字符串

日志内容 *

string ▼

def dw_process(msg): return m

🗑️

3. 在运行监控中查看日志。单击运行监控，选择运行日志，选择需要查看的集成流，即可看到对应的日志，目前支持查看历史日志和实时日志，历史日志可保留30天。

企业集成服务

项目管理

运行监控

自定义连接器

安全网关

成员管理

运行监控

概览

运行日志

定时任务管理

历史日志

实时日志

今天

昨天

近7天

选择日期

📅

默认项目 ▼

attribug ▼

全部流 ▼

按关键字搜索日志

搜索

项目名

应用名

Mapper

最近更新时间：2024-10-21 17:38:12

简介

RecordSet Mapper 组件用于操作 RecordSet，可以对输入的 RecordSet 进行转换，并生成一个新的 RecordSet。在转换的过程中，支持从其他数据源查找数据，并参与运算。

操作配置

参数配置

RecordSet Mapper 的配置界面包含三部分：

- 输入信息：会自动展示输入数据的字段和类型信息。
- 输出信息：需要用户自行配置输出的字段和类型信息。

- 逻辑映射：配置输入和输出字段间的映射关系。

Mapper ?

配置

备注

输入信息	新增	逻辑映射	新增	输出信息	新增
name string	⋮	○	○	response string	⋮
age int	⋮	○	○	age int	⋮
gender int	⋮	○	○	address string	⋮
address string	⋮	○			
firstName string	⋮	○			
lastName string	⋮	○			

映射逻辑配置

暂无映射逻辑，点击新增

配置输入信息

默认会自动识别 `msg.payload` 包含的字段以及字段类型进行展示。如果识别的结果不准确，用户也可以主动修改。



配置输出信息

需要用户自行配置输出数据中要包含的字段和类型。支持用户直接对输出字段进行赋值，支持赋值常量或者表达式：

- 常量：可以设置任意常量值。
- 表达式：在表达式里面可以用 `msg.payload` 处理所有输入的字段。



配置逻辑映射

用户需要指定输入信息和输出信息间的映射关系，支持两种方式：

- 直接赋值：通过连线直接连接输入字段和输出字段，输入和输出字段的类型需要一致。
- 使用映射逻辑节点：用户需要新增映射逻辑节点。对于每个映射逻辑节点，一般可以配置多个输入和输出，类似于一个函数的入参和出参。用户可以将 Mapper 组件输入信息中的字段连线到映射逻辑节点的输入，然后经过映射逻辑节点内部处理后，将映射逻辑节点的输出连线到 Mapper 组件输出信息中的字段进行赋值。映射逻辑配置可参考下文 [映射逻辑配置说明](#)。



映射逻辑配置

编辑配置

通过连接器查找: baidu-translate

流名称: transite

配置查找策略: 每次都重新查找

查询key信息

参数名	参数类型	默认值
id	int	
name	string	
score	float	

输出value信息

参数名	参数类型	默认值
out1	string	def dw_proc...
out2	int	433434

输入 message

message 属性	值
payload	继承上一个组件输出 message 中的 payload
error	空

attribute	空
variable	继承上一个组件输出 message 中的 variable 数据，在 RecordSet Filter 组件中可以操作 Variables，且不会影响输出 message 中的数据

输出

组件输出的 message 信息如下：

message 属性	值
payload	转换后的 RecordSet，其包含的字段为用户配置的输出字段
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	类型为 dict，继承上一个组件的 attribute
variable	继承上一个组件的 variable

映射逻辑配置说明

映射逻辑节点内部可以使用多种查找功能（从状态存储中查找、通过连接器查找、从其他流中查找）以及表达式处理功能。

从状态存储中查找

可以配置从 mapState、valueState、tableState 中查找数据，然后在映射节点内部进行处理并输出。下文以 mapState 为例进行说明。

Mapper ?

配置 备注

输入信息

新增

逻辑映射

新增

输出信息

新增

name string

age int

gender int

address string

firstName string

lastName string

statestore

key: string

out1: string

out2: string

out3: int

statestore

id string dataway

code string constant

name string

birthday date

interest string dataway

score float

sex string

company string

address string

使用逻辑映射节点

配置的摘要信息

映射逻辑配置

编辑配置

从状态存储中查找：mapState

表名称：student

配置查找策略：每次都重新查找

查询key信息

输出value信息

参数名	参数类型	默认值
key	string	

参数名	参数类型	默认值
out1	string	def dw_proc...
out2	string	
out3	int	123

通过连接器查找

支持通过连接器来查找第三方数据，然后在映射节点内部进行处理并输出。

Mapper ?

配置

备注

输入信息

新增

逻辑映射

新增

输出信息

新增

name string

score float

gender int

address string

firstName string

lastName string

connector

id: int

name: string

score: float

out1: string

out2: int

response string

使用映射节点

配置摘要

映射逻辑配置

编辑配置

通过连接器查找: baidu-translate

流名称: transite

配置查找策略: 每次都重新查找

查询key信息

参数名

参数类型

默认值

输出value信息

参数名

参数类型

默认值

id	int		out1	string	def dw_proc...
name	string		out2	int	433434
score	float				

从其他流中查找

新增映射逻辑

从其他流中查找

payload

配置查找策略

每次都重新查找

配置信息

配置流请求体

payload 值

f(x)

def dw_process(msg): return m

使用表达式来访问入参
比如, msg.payload['key1'],msg.payload['key2']

attribute 值

变量名	变量值	操作
attr1	string attr1	删除

+添加变量

也可以设置常量

variable 值

变量名	变量值	操作
var1	string var1	删除

+添加变量

确定

取消

Mapper ?

...

配置

备注

输入信息

新增

逻辑映射

新增

输出信息

新增

name string

age int

gender int

address string

firstName string

lastName string

key1: string

key2: string

speak: string

speak string

使用映射逻辑

配置摘要信息

映射逻辑配置

编辑配置

从其他流中查找： speak

配置查找策略：每次都重新查找

查询key信息

输出value信息

参数名	参数类型	默认值	参数名	参数类型	默认值
key1	string		speak	string	def dw_proc...
key2	string				

通过表达式配置

支持通过表达式来转换数据，即通过 Dataway 表达式来处理输入，然后输出。

Mapper ?

配置

备注

输入信息

新增

逻辑映射

新增

输出信息

新增

name string

score float

gender int

address string

firstName string

lastName string

expression

id: int

name: string

score: float

out1: string

out2: int

response string

映射逻辑配置

编辑配置

通过表达式配置

查询key信息

参数名

参数类型

默认值

id

int

name

string

score

float

输出value信息

参数名

参数类型

默认值

out1

string

def dw_proc...

out2

int

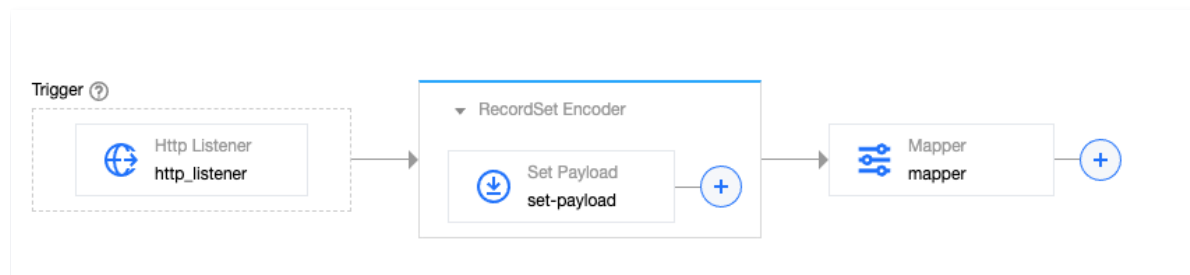
433434

如果在节点内部使用查找功能，用户还可以配置三种查找策略，合理使用查找策略可以提高执行效率。

查找策略	说明
每次都重新查找	对于输入的数据中的每一行，都重新执行一次新的查找。如果查找的结果随着输入的每行数据而变化，建议采用该策略。
只查找一次	无论输入的数据有多少行，都只执行一次查找，后续查找都继续使用第一次查找的结果。如果查找的结果不随着输入数据变化，建议采用该策略。
周期性查找	可以配置一个查找周期，仅当两次查找的时间间隔超过配置的查找周期时，才会重新执行查找，否则，继续使用上一次查找的结果。如果查找的结果是周期性更新的，建议采用该策略。

案例

针对一个学生表进行转换，示例图如下：



1. 用 RecordSet Encoder 组件创建一个学生表，包含 `name, age, address, score, adult` 字段，然后在 RecordSet Encoder 组件里面添加 SetPayload 组件，用表达式在学生表中添加数据。

RecordSet ... ?

配置 备注

基本配置

输出Schema

序号	字段类型	字段名	显示名称	操作
1	string	name	name	删除
2	int	age	age	删除
3	string	address	address	删除
4	float	score	score	删除
5	bool	adult	adult	删除

+添加变量

Dataway表达式 ?

```

1 def dw_process(msg):
2   return [
3     ['张三', 10, '陕西省西安市', 99.5, False],
4     ['张三', 20, '湖北省荆州市', 61.2, True],
5     ['张三', 25, '湖北省洪湖市', 71.1, True],
6     ['张三', 30, '湖北省赤壁市', 30.33, True],
7     ['张三', 40, '湖北省襄阳市', 80.33, False],
8     ['张三', 80, '湖北省襄阳市', 40.33, True],
9     ['张三', 90, '湖北省襄阳市', 30.33, False],
10    ['李四', 70, '北京省长江市', 10.33, True],
11    ['王五', 80, '广东省深圳市', 70.33, True],
12    ['王五', 30, '湖南省长沙市', 73.33, True],
13    ['王五', 40, '湖南省长沙市', 23.33, False],
14  ]
  
```

2. 用 Mapper 组件进行转换，生成一个新的学生表，包含 `code, name, company, homeAddress, isAdult` 5个字段，Mapper 组件的整体配置视图如下：

Mapper ?

配置 备注

输入信息

```

::: name string
::: age int
::: address string
::: score float
::: adult bool

```

自动填充上一个组件输出结果中的字段

逻辑映射

使用表达式转换

直连

```

expression ***
key1: string
key2: int
key3: float
out1: string
out2: string

```

输出信息

直接设置常量

手动配置输出字段

直接配置表达式，不用连线

```

::: code string
::: name string
::: company string constant
::: homeAddress string dataway
::: isAdult bool

```

映射逻辑配置

通过表达式配置

查询key信息

参数名	参数类型	默认值
key1	string	
key2	int	
key3	float	

输出value信息

参数名	参数类型	默认值
out1	string	def dw_proc...
out2	string	def dw_proc...

3. 用表达式转换节点来处理输入信息中的 name、age、score，然后赋值给输出信息中的 code、name，配置信息如下：

新增映射逻辑

通过表达式配置

配置查询Key信息

参数名	参数类型	默认值
key1	string	请输入默认值
key2	int	请输入默认值
key3	float	请输入默认值

+ 添加字段

配置输出value信息

参数名	参数类型	输出结果
out1	string	表达式 def dw_process(msg):
out2	string	表达式 def dw_process(msg):

+ 添加字段

```

def dw_process(msg):
    return msg.payload["key1"] + str(msg.payload["key2"]) + str(msg.payload["key3"])

```

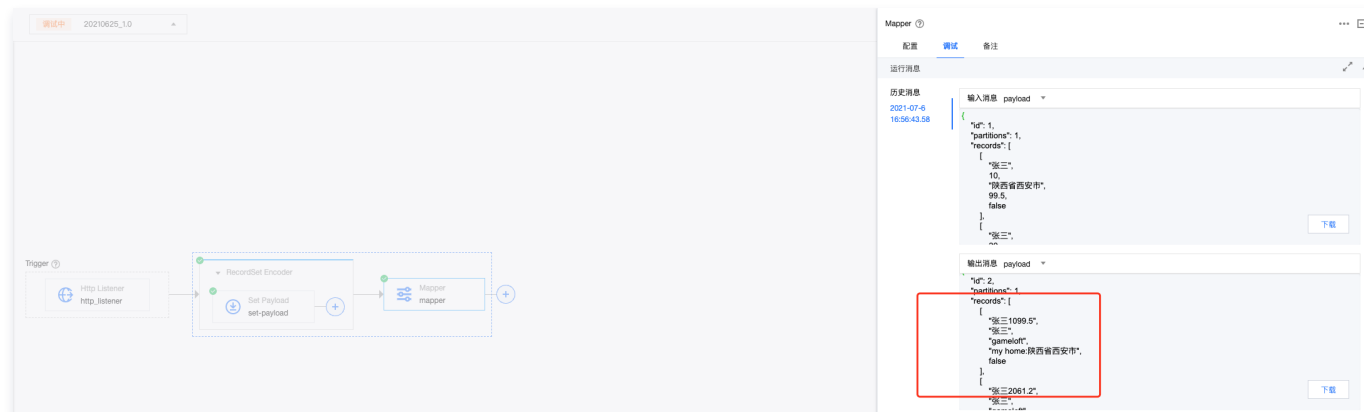
```

def dw_process(msg):
    return msg.payload["key1"]

```

确定 取消

- 输出信息中的 `company` 直接设置为常量 `gameloft`。
- 输出信息中的 `homeAddress` 直接设置为表达式 `return "my home: "+msg.payload["address"]`。
- 输入信息中的 `adult` 直连输出信息中的 `isAdult`。
- 在调试模式下进行单元测试，并查看结果。



```
[
  [
    "张三1099.5",
    "张三",
    "gameloft",
    "my home:陕西省西安市",
    false
  ],
  [
    "张三2061.2",
    "张三",
    "gameloft",
    "my home:湖北省荆州市",
    true
  ],
  [
    "张三2571.1",
    "张三",
    "gameloft",
    "my home:湖北省洪湖市",
    true
  ],
  [
    "张三3030.33",
    "张三",
    "gameloft",
    "my home:湖北省赤壁市",
    true
  ],
  [
    "张三4080.33",
    "张三",
    "gameloft",
    "my home:湖北省襄阳市",
    false
  ],
  [
    "张三8040.33",
    "张三",
    "gameloft",
    "my home:湖北省襄阳市",
    true
  ]
]
```

```
],  
[  
  "张三9030.33",  
  "张三",  
  "gameloft",  
  "my home:湖北省襄阳市",  
  false  
],  
[  
  "李四7010.33",  
  "李四",  
  "gameloft",  
  "my home:北京省长江市",  
  true  
],  
[  
  "王五8070.33",  
  "王五",  
  "gameloft",  
  "my home:广东省深圳市",  
  true  
],  
[  
  "王五3073.33",  
  "王五",  
  "gameloft",  
  "my home:湖南省长沙市",  
  true  
],  
[  
  "王五4023.33",  
  "王五",  
  "gameloft",  
  "my home:湖南省长沙市",  
  false  
]  
]`
```

Parallel Foreach

最近更新时间：2023-06-08 14:47:56

简介

Parallel Foreach 用于并行执行任务，Parallel Foreach 对数据集上的元素并行执行相同的处理逻辑，并行数目依赖于用户配置的最大并行数，当元素数目小于等于最大并行数时，并行数等于元素数目，当元素数目大于最大并行数时，并行数为用户配置的最大并行数。处理完成后，每个元素的处理结果按照原始顺序输出到 message 的 payload 中。Parallel Foreach 一般用于批量数据处理的场景，例如：批量查询、批量导入数据等。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
数据集	string、list、dict、int	待遍历的数据： - 当类型为 string 时，遍历字符串的每个字符。 - 当类型为 list 时，遍历 list 的每个元素。 - 当类型为 dict 时，遍历 dict 中的 value。 - 当数据集类型为 int 时，例如3，实际遍历的数据集为[0,1,2]。	是	无
最大并行数	int	并行执行的任务数，取值范围为2~8。	是	4
计数器	string	计数器是一个变量，该变量存储当前的迭代次数，从0开始，此处填入变量名称，msg.vars.get('#计数器变量#') 即可使用；例如：当计数器变量使用默认值 counter 时，第1次循环，msg.vars.get('counter') 值为0，第2次循环，msg.vars.get('counter') 值为1。	否	count er
根信息	string	根信息同样是一个变量，此处填入变量名称，根信息中保存主流的 message 信息，msg.vars.get('#根信息名称#').payload 即可访问主流的 payload 数据。当使用默认值 rootMessage 时，使用 msg.vars.get('rootMessage').payload 即可在 Parallel ForEach 的子流中访问主流的 payload 数据。	否	rootM essag e

配置界面

Parallel For... ?

配置 输入 输出 备注

...

基本配置

数据集 •

点击输入DataWay表达式

循环遍历的数据集对象

最大并行数 •

4

同时执行的任务数,系统范围:[2,8]

计数器 •

counter

循环遍历的计数器

根信息 •

rootMessage

用于指定循环遍历的父级流消息对象

输入到子流中的 message

message 属性	值
payload	数据集集中的元素，例如： 待处理的数据集为[1,2,3]，第一个并行任务中，子流中 payload 的数据为1，第二个并行任务的 payload 为2。

	待处理的数据集为 dict 类型，{"key":"key1", "value":"value1"}，第一个并行任务中，子流中的 payload 为 key1，第二个并行任务的 payload 为 value1。
error	空。
attribute	空。
variable	包含两个变量，一个是计数器，一个是根信息，若用户使用默认值，可使用表达式 msg.vars.get('counter') 和 msg.vars.get('rootMessage') 访问。

输出

Parallel Foreach 的输出结果中，不会包含处理逻辑中使用的 variable 变量，最终输出的只有 payload 里的数据，输出的 payload 是个 list 类型变量，里边包含原始数据集中每个元素迭代处理的结果，顺序与原始数据集一致。attribute 的值继承自上一个组件。

组件输出的 message 信息如下：

message 属性	值
payload	list 类型，包含了每个数据的处理结果，输出顺序与原始的输入顺序一致。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息。
attribute	继承 Parallel ForEach 上一个组件的 attribute 信息。
variable	继承 Parallel ForEach 上一个组件的 variable 信息。

输出样例

输出消息 payload ▾

```
[
  2,
  3,
  4,
  5
]
```

下载

案例

本案例中，我们使用 Parallel Foreach 组件对列表中的数据统一进行加1操作，原始的数据集为[1,2,3,4]。

1. 添加 Parallel Foreach 组件，配置数据集[1,2,3,4]，并设置并行度为4。

Parallel For...

配置
备注

基本配置

数据集

def dw_process(msg): return [1,2,3,4]

循环遍历的数据集对象

最大并行数

int
4

同时执行的任务数,系统范围:[2,8]

计数器

counter

循环遍历的计数器

根信息

rootMessage

用于指定循环遍历的父级流消息对象

2. 在 Parallel Foreach 中拖入 Set Payload 组件，对数据集执行加1操作。

Set Payload

配置
备注

配置 / 编辑

Dataway表达式

1 def dw_process(msg):
2 return msg.payload+1

自动联想
payload
payload

3. 输出结果。

输出消息 payload

[
2,
3,
4,
5
]

下载

Raise Error

最近更新时间：2023-06-08 14:47:56

简介

Raise Error 组件用来抛出异常，中断流的执行。该组件可以单独使用，也可以搭配 Try 组件使用。单独使用时，命中 Raise Error，集成流会抛出错误，中断执行。搭配 Try 组件使用时，可以在 Catch 中捕获 Raise Error 定义的异常，然后执行 Catch 中配置的子流。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
错误类型	string	用户自定义错误类型	是	无
错误描述	tring	错误的描述信息	是	无

配置界面

Raise Error ?

配置 输入 输出 备注

...

基本配置

错误类型 *

请输入错误类型

错误的类型

错误描述 *

string

输入字符串

错误的描述,支持表达式输入

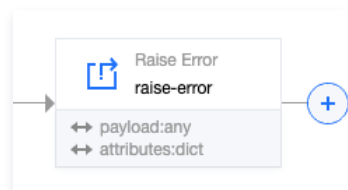
输出

组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload。
error	error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	继承上个组件的 variable 信息。

案例

1. 添加 Raise Error 组件。



2. 填入错误类型和对应的错误描述。

Raise Error

?

...

配置

备注

基本配置

错误类型 *

requestError

错误描述 *

string

request failed

f(x)

3. 输出。

输出消息 error

{
 "Code": "requestError",
 "Description": "request failed"
}

下载

Remove Variable

最近更新时间：2023-06-08 14:47:56

简介

Remove Variable 组件与 Set Variable 组件的作用相反，Remove Variable 组件的作用是删除 message 中的一个指定变量。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
变量名称	string	要移除的变量名称	是	无

配置界面

Remove Variable ?

配置 输入 输出 备注

基本配置

变量名 *

请输入变量名

变量的名称

输出

输出的 message 中，不再包括删除的变量。组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	上个组件的 variable 信息中去除删掉的变量。

输出示例

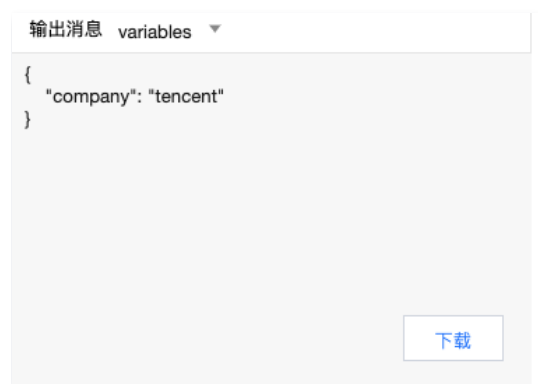
输入到 Remove Variable 中的变量如下：

输入消息 variables ▾

```
{
  "company": "tencent",
  "name": "ipaas"
}
```

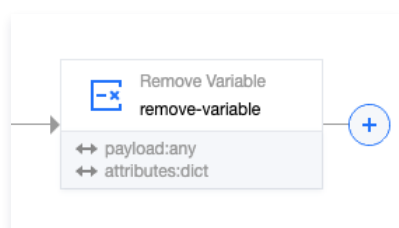
下载

使用 Remove Variable 删除变量 name 后，如下所示：



案例

1. 添加 Remove Variable 组件。



2. 填入要移除的变量名称。



RecordSet Encoder

最近更新时间：2023-06-08 14:47:56

简介

RecordSet Encoder 用于将普通的数据类型组装为 RecordSet 数据类型，用于后续的批量数据处理。在 RecordSet Encoder 中可以配置子流，返回一个普通类型的数据列表。同时在组件配置中指定需要构造的 RecordSet 数据的 schema 和分片数，系统会将根据这些配置信息对子流返回的数据列表进行解析，从而得到一个支持多分区的 RecordSet 数据对象。

操作说明

参数配置

RecordSet Encoder 组件的配置界面如下：

配置

备注

基本配置

输出Schema

序号	字段类型	字段名	显示名称	操作
1	int	Name	aaa	删除

+添加变量

高级配置

输出持久化

boolfalse

分区数

def dw_process(msg): return 3

配置参数说明：

参数	数据类型	描述	是否必填	默认值
输出 schema	图形化配置	需要输出的 RecordSet 数据的字段结构信息，需要指定字段名（必填）、字段类型（必填）、显示名称（非必填）	是	无
输出持久化	bool	用于指定是否需要将输出的 RecordSet 进行持久化存储，若选择 true，则会在该 RecordSet 第一次计算后缓存计算结果，后续对该数据的读取将无需重复计算	否	false
分区数	int	用于指定输出的 RecordSet 的分区数，在实际获取 RecordSet 数据时，有多少个分区则会执行多少次子流，从而为每个分区分别获取数据	否	1

子流中的输入 message

RecordSet Encoder 是嵌套组件，可以指定子流程，子流程的输入 message 内容规定如下：

message 属性	值
payload	当前分区的索引，例如：指定3个分区，则分别会以0、1、2作为 payload 来调用子流
error	空
attribute	空

variable	继承主流中的 variable 数据，同时新增两个变量，分别为：计数器、根信息，若用户使用默认值，可使用表达式 <code>msg.vars.get('counter')</code> 和 <code>msg.vars.get('rootMessage')</code> 访问。在子流中操作 Variables 将不会影响主流中的数据
----------	---

输出

RecordSet Encoder 组件输出的 message 中，保留前一个组件的 Variables 和 attributes 信息，只有 payload 会被设置为输出的 RecordSet 数据对象。组件输出的 message 信息如下：

message 属性	值
payload	构造得到的 RecordSet 对象
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承前一个组件输出的 attribute 信息
variable	继承前一个组件输出的 variables 信息

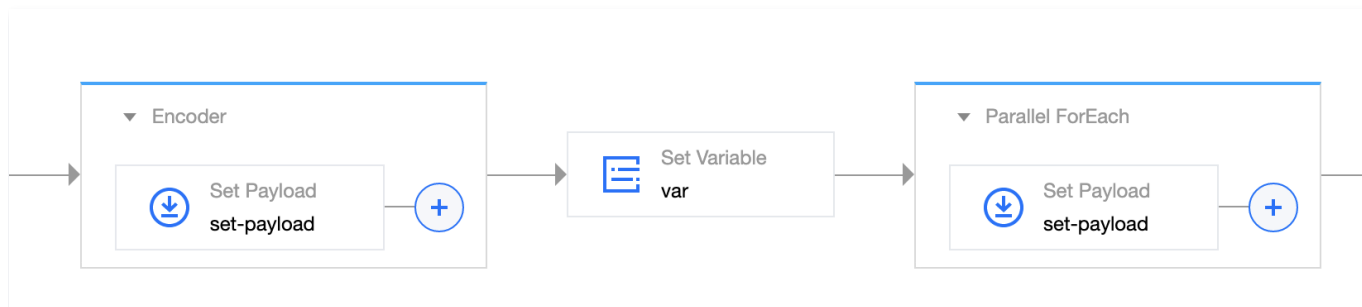
执行原理

分区数

RecordSet 中的数据分区是为了充分利用系统的并行计算能力而设计的，在计算时，不同的分区之间可以进行并行计算。在 RecordSet Encoder 中，可以设置分区数为[1,100]范围中的任意整数，系统将会输出具有指定分区数的 RecordSet 对象。

延迟计算

由于 RecordSet 数据类型本身具有的延迟计算的特性，RecordSet Encoder 组件中的子流会在后续流程尝试获取其输出的 RecordSet 的内容时才会真正开始执行，如下集成流：



在开始执行 Parallel ForEach 节点时，才会开始触发 RecordSet Encoder 中的子流程的执行。

持久化缓存

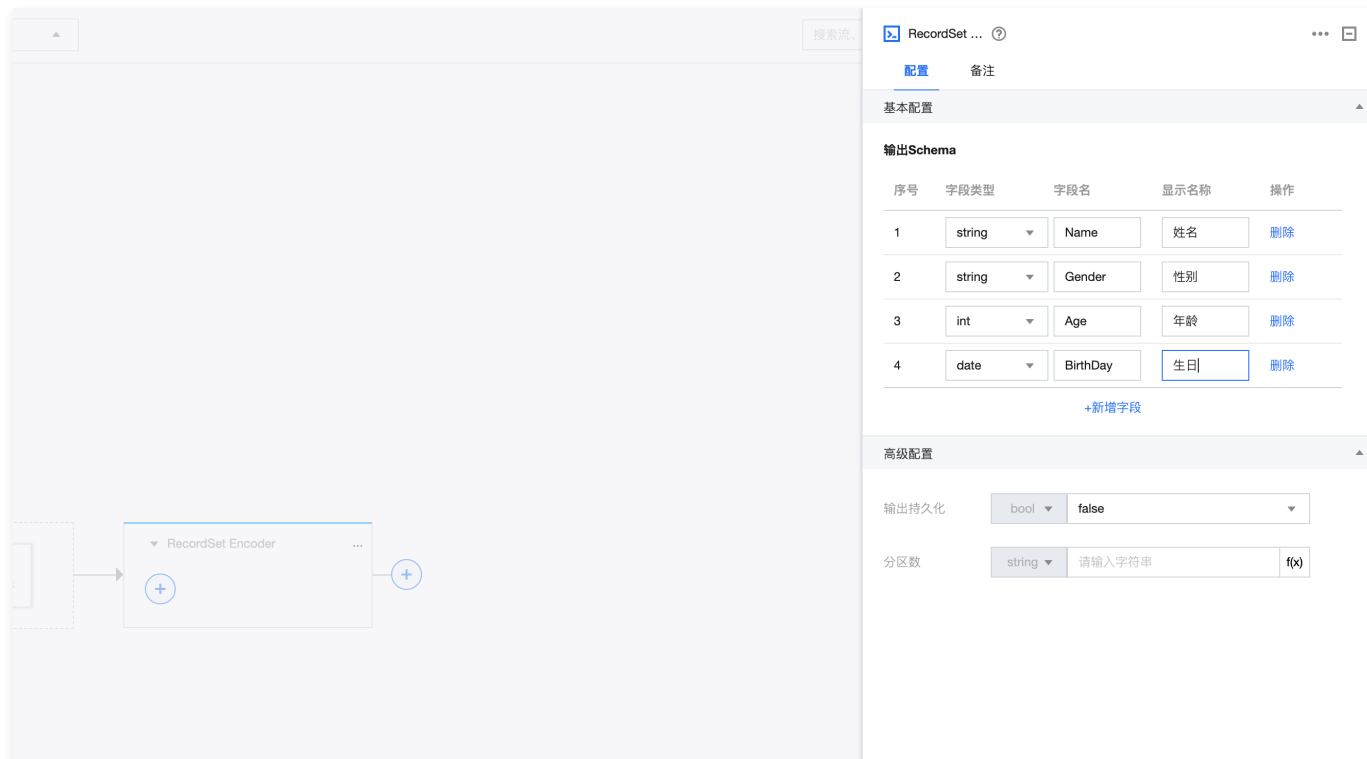
在设计上，RecordSet 中的数据是可重复读取的，其实现的机制是每次都会触发 RecordSet 中计算逻辑，得到新的输出。在 RecordSet 的计算复杂度较高的情况下，重复发起计算流程可能会导致性能低下。持久化缓存主要就是为解决这一问题而设计的。

若开启持久化缓存选项，系统将会在 RecordSet 第一次计算后将结果持久化到对象存储上，后续的读取将会直接从存储中加载而不会触发重复计算，因此能很大程度降低系统性能负载。

使用案例

使用 RecordSet Encoder 组件构造一个具有三个分区的 RecordSet。

1. 添加 RecordSet Encoder 组件，设置输出 schema。



RecordSet Encoder 配置

基本配置

输出Schema

序号	字段类型	字段名	显示名称	操作
1	string	Name	姓名	删除
2	string	Gender	性别	删除
3	int	Age	年龄	删除
4	date	BirthDay	生日	删除

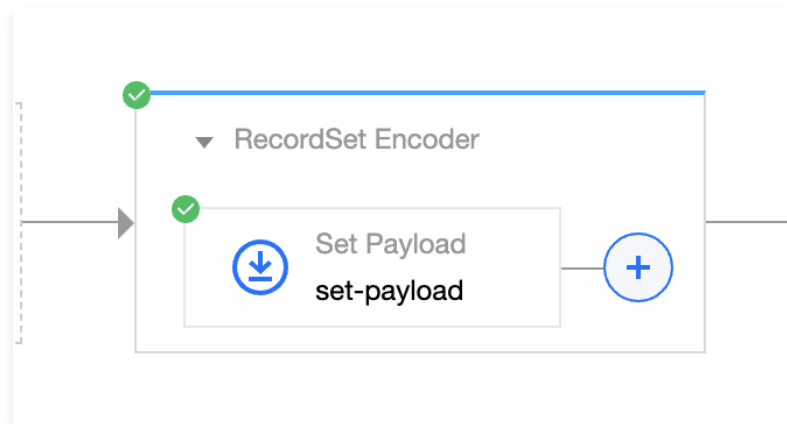
[+新增字段](#)

高级配置

输出持久化: bool ☐ false

分区数: string 请输入字符串 f(x)

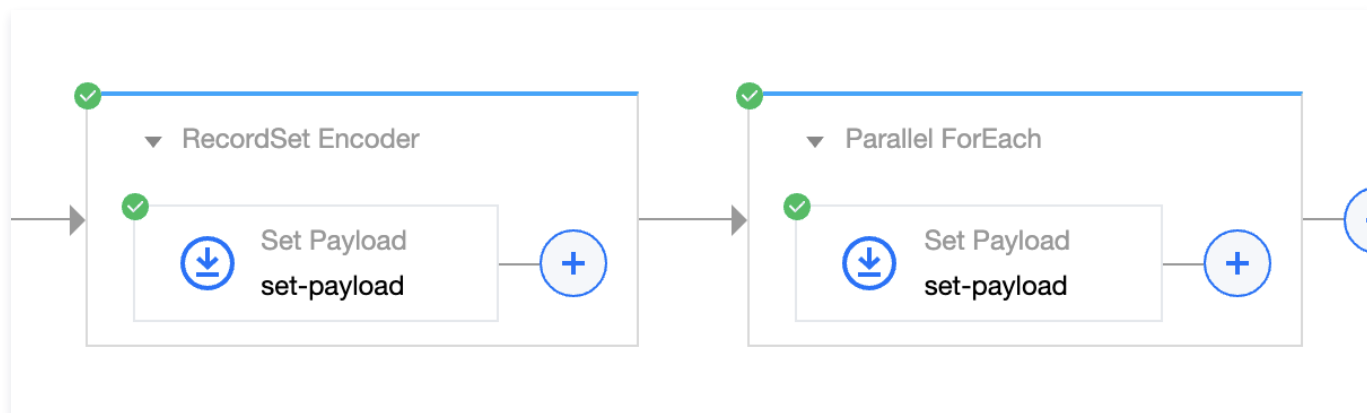
2. 配置子流，将子流的 payload 设置为需要构造的数据内容，子流配置：



其中 set-payload 的配置如下：

```
def dw_process(msg):
    return [{
        "Name": "fuz",
        "Gender": '男',
        "Age": 20,
        "BirthDay": datetime.datetime.now().date(),
    }, {
        "Name": "yat",
        "Gender": "女",
        "Age": 20,
        "BirthDay": datetime.datetime.now().date(),
    }]
}
```

3. 在下游通过 ForEach 或者 Parallel ForEach 组件遍历输出的 RecordSet，并查看结果。



4. 通过 debug 模式运行并查看，可以得到如下结果。

Parallel ForEach

配置 调试 备注

运行消息

历史消息

2021-06-25 19:25:17.14

2021-06-25 19:20:44.376

输入消息 payload

```
{
  "id": 4,
  "partitions": 3,
  "records": [
    [
      "fuz",
      "男",
      20,
      "2021-06-25"
    ],
    [
      "fuz",
      "男",
      20,
      "2021-06-25"
    ]
  ]
}
```

输出消息 payload

```
[
  [
    "fuz",
    "男",
    20,
    "2021-06-25"
  ],
  [
    "yat",
    "女",
    20,
    "2021-06-25"
  ],
  [
    "yat",
    "女",
    20,
    "2021-06-25"
  ]
]
```

Mock ?

以下变量值，请至少配置1项

Scatter Gather

最近更新时间：2023-06-08 14:47:56

简介

Scatter Gather 支持并行执行多个任务，该组件中，有两个核心配置，一个是并行数，目前支持的并行数是2~8个，另一个是 worker，每个 worker 中都可以配置一个待执行的任务。当 worker 数目和并行数不一致时，实际的并行数目是两者中的较小值。Scatter Gather 以上一个节点输出的 message 作为输入，并行执行 worker 中配置的任务。最后的输出以字典呈现，字典的 key 是 worker 的下标，从1开始，value 是对应 worker 执行完成后的 payload，message 中的 attributes 和 variables 都会丢弃。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
最大并行数	int	并行执行的任务数，取值范围为2~8。	是	4
根信息	string	根信息同样是一个变量，这里填入变量名称，根信息中保存主流的 message 信息，msg.vars.get('#根信息名称#').payload 即可访问主流的 payload 数据。当使用默认值 rootMessage 时，使用 msg.vars.get('rootMessage').payload 即可在子流中访问主流的 payload 数据。	否	rootMessage

配置界面

Scatter Gat... ?

配置 输入 输出 备注

...

基本配置

最大并行数

4

同时执行的任务数,系统范围:[2,8]

根信息

rootMessage

用于指定循环遍历的父级流消息对象

输入子流中的 message

message 属性	值
payload	继承 Scatter Gather 上一个组件的 payload 信息。
error	空。
attribute	空。
variable	包含两个变量，一个是计数器，一个是根信息，若用户使用默认值，可使用表达式 msg.vars.get('counter') 和 msg.vars.get('rootMessage') 访问。

输出

worker 的输出结果中不会包含处理逻辑中使用的 variable 变量，最终输出的只有 payload 里的数据，Scatter Gather 的 payload 是汇总每个 worker 的 payload 结果，attribute 和 variables 保留上一个节点的输出。如图所示："1"表示 worker1 的输出，"2"表示 worker2 的输出。



组件输出的 message 信息如下：

message 属性	值
payload	dict 类型，key 是 work 的编号，从"1"开始，value 是 work 执行完成后的结果。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承 Scatter Gather 上一个组件的 attribute 信息。
variable	继承 Scatter Gather 上一个组件的 variable 信息。

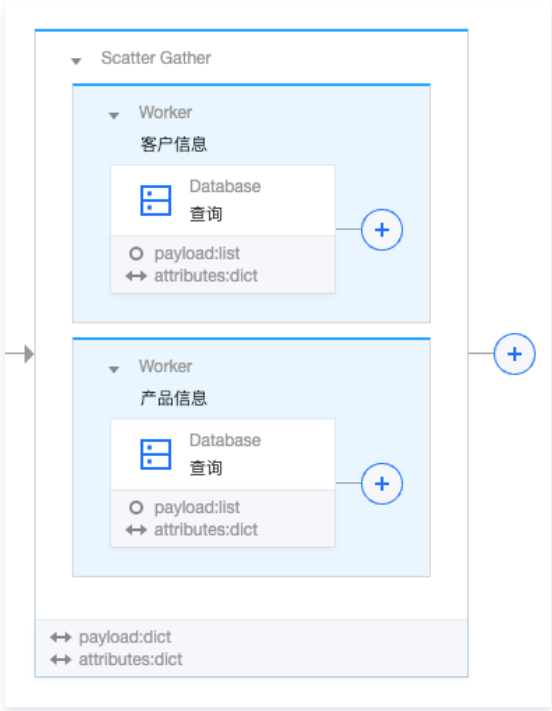
案例

当需要并行执行不同的任务时，使用 Scatter Gather 组件比较适合，假设需要根据用户订单数据，去查询客户信息及产品信息，则可以配置两个 worker，一个执行客户信息查询，一个执行产品信息查询。

1. 添加 Scatter Gather 组件，设置最大并行度为2。



2. 第一个 worker 配置客户信息查询，第二个 worker 配置产品信息查询，在 Scatter Gather 的输出中，即可看到客户和产品信息。



3. 查询结果，字典中 key 为“1”的元素，保存客户信息，key为“2”的元素，保存产品信息。

```
{
  "1": {
    "msg": "成功",
    "code": 200,
    "data": {
      {
        "crmId": 4569221968,
        "followUserId": 14151417,
        "groupId": 0,
        "name": "test",
        "call": "",
        "gender": "未知",
        "birthday": "",
        "channel": ""
      }
    }
  },
  "2": {
    "code": 200,
    "msg": "成功",
    "data": {
      {
        "id": "DD6202104074138457160",
        "userId": 14151417,
        "userName": "测试",
        "createTime": "2021-04-07 11:29:44",
        "changeTime": "2021-04-07 11:29:44",
        "projectName": "test_订单(2021-04-07 11:29)",
        "status": 2,
        "crmId": "4569221968"
      }
    }
  }
}
```

Scheduler

最近更新时间：2023-06-08 14:47:56

简介

Scheduler 是定时器，用于按设定的规则定时触发一条流。图形化 Scheduler 支持三种触发模式：单次触发、周期触发和 Cron 表达式。

- 单次触发模式：支持选择多个指定的时间点触发。
- 周期触发模式：按照固定的周期触发。
- Cron 表达式模式：配置中包含一条或多条 Cron 规则，当任意一条 Cron 规则与当前时间匹配时，该定时器所在的集成流将被触发。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
触发模式	Int	设置触发模式：单次触发，周期触发，Cron 表达式	是	0（Cron 表达式模式）
Cron表达式	string	设置触发规则，例如每分钟触发1次等	是	无
时区	string	指定时区	是	UTC+08:00
排队调度	bool	若勾选当前有未结束的作业时跳过本次执行，则为 true	否	false

Scheduler 包含一条或多条 cron 规则，当存在多条规则时，使用“\r”分隔，Cron 表达式的配置规则如下：

参数	含义	取值范围
seconds	秒	0 – 59
minutes	分钟	0 – 59
hours	小时	0 – 23
days	日期（选填，默认每天）	1 – 31
months	月份（选填，默认每月）	1 – 12
weekdays	星期几（选填，默认不指定）	1 – 7
years	年份（选填，默认每年）	1970 – 2099

在配置 Cron 表达式时，可使用以下运算符：

运算符	释义
*	表示任意匹配，例如： <code>hours="*"</code> 表示每个小时。
–	表示范围，例如： <code>weekdays="1-5"</code> 表示周一至周五。
,	表示并列，例如： <code>months="1,3,5,7,8,10,12"</code> 表示大月。
/	表示增量，例如： <code>hours="8/2"</code> ，表示8点开始，每两个小时。
L	表示最后一个，例如： <code>weekdays="6L"</code> ，表示当月最后一个周六。
?	表示不指定。此处有一个约束，即年月日和星期几中至少要有一个为不指定。该约束是为了避免冲突，例如：同时指定2020年2月20日（原本周四）和周三，默认星期几是不指定的。

配置界面

Scheduler ?

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

Cron表达式 *

请输入Cron表达式

访问更多Cron信息

指定cron表达式,由空格分割,主要结构为: 秒 分 小时 日期 月份 星期 (年, 可选)

高级配置

时区 *

UTC+08:00

指定时区

☐

若当前有未结束的作业时跳过本次执行

输出

Scheduler 作为 trigger 组件，是集成流的第一个组件，Scheduler 会生成一个空的 message 消息，触发集成流的运行。
组件输出的 message 信息如下：

message 属性	值
payload	空，没有值存在。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	空，没有值存在。
variable	空，没有值存在。

案例

单次触发模式

2025年1月1日0时0分0秒触发一次。

 Scheduler ?

...

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

触发时间 *

2025-01-01 00:00:00



+ 新增

高级配置

时区 *

UTC+08:00

▼

指定时区

☐

若当前有未结束的作业时跳过本次执行

周期触发模式

- 每5秒触发一次。

 Scheduler ?

...

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

触发周期 *

每

5

秒 ▼

触发

高级配置

时区 *

UTC+08:00

▼

指定时区

☐

若当前有未结束的作业时跳过本次执行

- 每1分钟的0秒触发一次。

Scheduler ?

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

触发周期 *

每

1

分钟

的

00

触发

高级配置

时区 *

UTC+08:00

指定时区

☐ 若当前有未结束的作业时跳过本次执行

Cron表达式模式

- 每5分钟触发一次。

Scheduler ?

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

Cron表达式 *

0 0/5 * * * ?

访问更多Cron信息

指定cron表达式,由空格分割,主要结构为: 秒 分 小时 日期 月份 星期 (年, 可选)

高级配置

时区 *

UTC+08:00

指定时区

☐ 若当前有未结束的作业时跳过本次执行

- 每天8点触发。

 Scheduler 

配置

备注

基础配置

配置模式 *

单次触发

周期触发

Cron表达式

Cron表达式 *

0 0 8 * * ?

[访问更多Cron信息](#) 

指定cron表达式,由空格分割,主要结构为: 秒 分 小时 日期
月份 星期 (年, 可选)

高级配置

时区 *

UTC+08:00



指定时区

☐ 若当前有未结束的作业时跳过本次执行

Set Payload

最近更新时间：2023-06-08 14:47:56

简介

Set Payload 可以设置 message 中的 payload 属性，支持表达式和字面量两种输入形式，如果是字面量形式，需要先选择数据类型，再在输入框中填入字面量。如果需要输入表达式，选择 any 类型，写入表达式即可。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
值	any	payload 中要保存的数据。	是	无

输出

组件输出的 message 信息如下：

message 属性	值
payload	用户填入的数据。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	继承上个组件的 variable 信息。

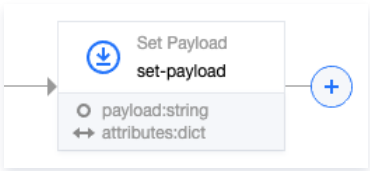
payload

"hello"

下载

案例

1. 添加 Set Payload 组件。



2. 填入要设置的数据。

Set Payload ?

配置备注

基本配置

值 *

string

ipaas

f(x)

Set Variable

最近更新时间：2023-06-08 14:47:56

简介

Set Variable 的作用是声明一个变量，并保存在 message 的 variables 中，后续节点可通过 msg.vars.get('name') 形式引用该变量。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
变量名	string	变量名称。	是	无
变量值	any	变量的具体值。	是	无

配置界面

Set Variable ?

配置 输入 输出 备注

基本配置

变量名 *

company

变量的名称

变量值 *

string

tencent

输出

对 variables 变量的引用，需要使用表达式：msg.vars.get('company')。组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	上个组件的 variable 信息加上当前组件添加的变量。

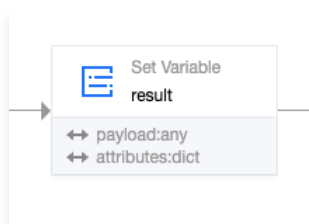
variables

{
 "company": "tencent"
}

下载

案例

1. 添加 Set Variable 组件。



2. 在“变量名”处填入变量名称，“变量值”处填入要保存的值。



Synchronized

最近更新时间：2023-06-08 14:47:56

简介

Synchronized 是一个分布式锁。对并发处理的请求进行加锁保护，可应用在对资源互斥的访问场景中。目前加锁保护支持流级别加锁。在不同的项目中，即使“加锁对象”名称相同，也是不同的两把锁，但在同一个项目中，如果“加锁对象”名称相同，则只定义了一把锁。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
加锁对象	string	声明一个锁对象。	是	无
最大锁持有时间	int	锁最大占有时间，目前最大支持3600秒。	是	3600
最大锁等待时间	int	最长等待时间，目前最长等待时间为3600秒。	是	3600

配置界面

Synchronized ?

配置 输入 输出 备注

...

□

基本配置

加锁对象

string

输入字符串

待加锁的资源对象标识

最大锁持有时间

3600

锁状态持续时间，超时自动释放锁，单位秒

最大锁等待时间

3600

未加锁请求等待时间，设置0为请求立即失败，单位秒

输入到子流中的 message

message 属性	值
payload	继承 Synchronized 上一个组件的 payload。
error	空。
attribute	继承 Synchronized 上一个组件的 attribute 信息。
variable	继承 Synchronized 上一个组件的 variable 信息。

输出

组件输出的 message 信息如下：

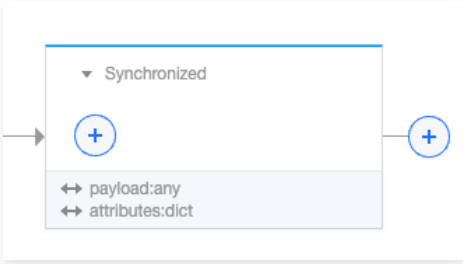
message 属性	值
payload	继承子流输出的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错

	误类型，“Description” 字段表示错误具体信息。
attribute	继承子流输出的 attribute 信息。
variable	继承子流输出的 variable 信息。

案例

在资源互斥的场景中，可以使用 Synchronzied 对资源进行加锁，例如：库存扣减操作，对库存查询和库存扣减加锁，防止超卖。

1. 添加 Synchronzied 组件，声明锁对象 stock。



Synchronized ?

配置

备注

基本配置

加锁对象 *

请选择

stock

f(x)

最大锁持有时 * 间

请选择

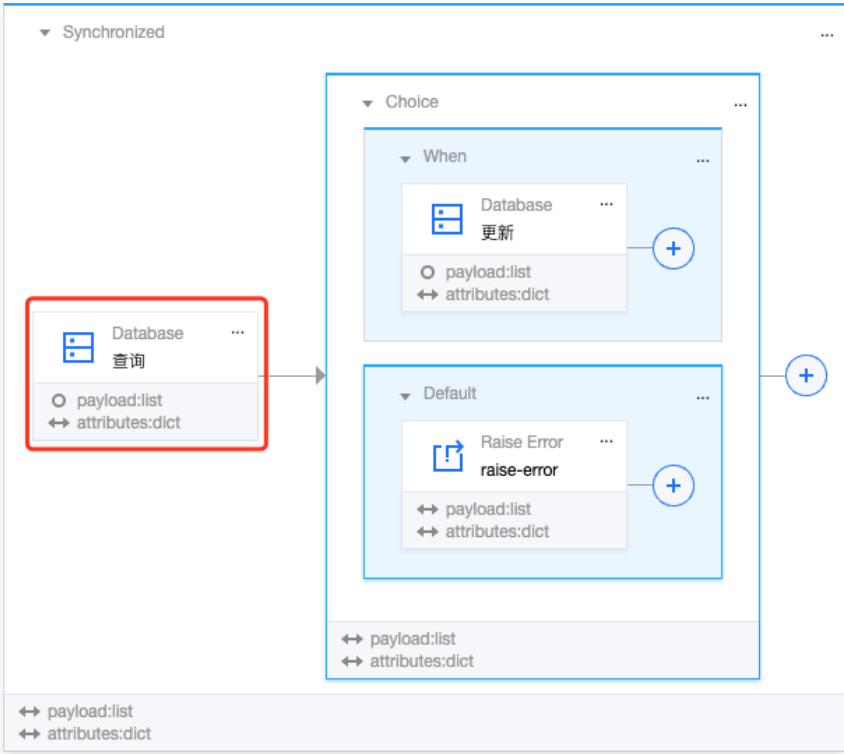
3600

最大锁等待时 * 间

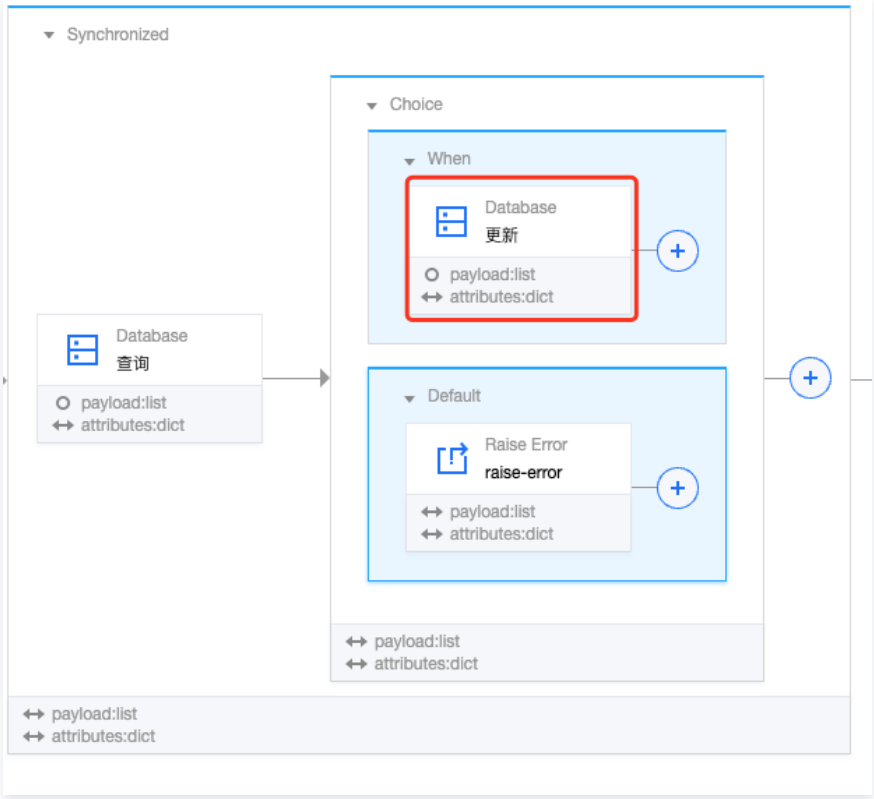
请选择

3600

2. 在 Synchronzied 中配置子流，添加 Database 组件，查询库存。



3. 库存足够时，进行扣减操作。



Transform

最近更新时间：2023-06-08 14:47:57

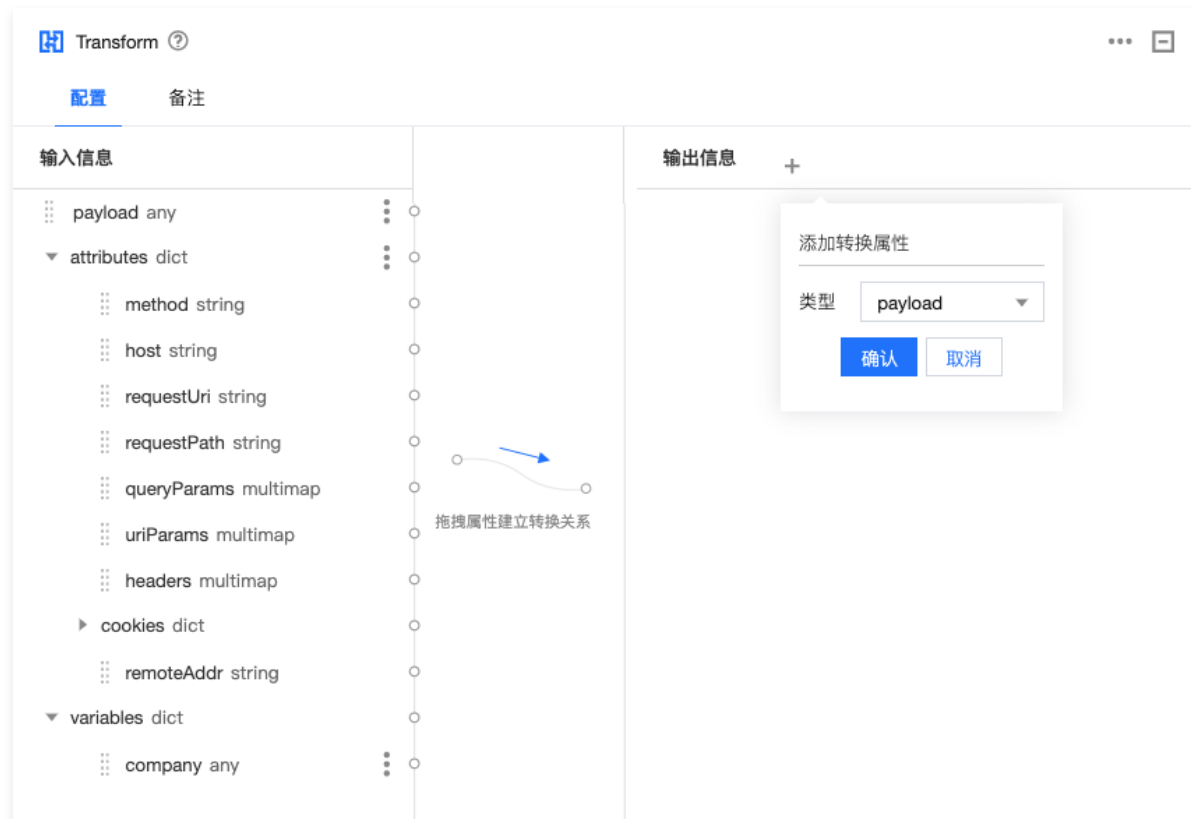
简介

Transform 组件可以对 message 消息进行数据编排和格式转换，支持 payload、attribute 和 variable 的修改，在使用 Transform 编辑 payload 时，需要先在 dataType 中定义要转换的数据类型，然后在 Transform 中绑定该类型，最后对输入消息进行编排，映射到绑定的数据类型上。

操作配置

参数配置

1. 确定要转换的类型，目前可选的有 payload，attributes，variables。



2. 创建新的数据类型。

Transform ?

配置

备注

输入信息

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

company any

输出信息

payload x +

any

string

int

float

bool

decimal

datetime

date

time

绑定自定义数据类型

添加自定义数据类型

绑定数据类型

编辑自定义数据类型

×

名称

Custom_Type

数据类型

Entity

请选择一种Entity类型

JSON

1

{

2

"name": "a",

3

"age": 10

4

}

预览

▼ Custom_Type <entity>
 name string
 age int

确定

取消

3. 绑定数据类型。

Transform ?

配置

备注

输入信息

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

age int

拖拽属性建立转换关系

输出信息

payload × +

any

string

int

float

bool

decimal

datetime

date

time

绑定自定义数据类型

添加自定义数据类型

绑定数据类型

4. 拖动字段，进行匹配。

Transform ?

配置

备注

输入信息

payload any

attributes dict

- method string
- host string
- requestUri string
- requestPath string
- queryParams multimap
- uriParams multimap
- headers multimap

cookies dict

- remoteAddr string

variables dict

- age int

输出信息

payload x +

Custom_Type <entity>

- name string
- age int

版权所有：腾讯云计算（北京）有限责任公司

第137 共732页

5. 当图形转换无法满足业务需求时，可使用脚本输入进行补充。

Transform ?

配置

备注

输入信息

输出信息 payload x +

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

age int

Dataway表达式 ?

process(msg):
{ 'name': msg.attrs['method'], 'age': msg.

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

attributes

attrs

vars

vars["age"]

图形转换

脚本输入

输出

组件输出的 message 信息如下：

message 属性	值
payload	如果“输出信息”中添加 payload，输出为 payload 中的执行结果，否则继承上一个组件的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	类型为 dict，如果“输出信息”中添加 attributes，输出为 attributes 的执行结果，否则继承上一个组件的 attribute。
variable	如果“输出信息”中添加 variables，输出为上一个组件的 variable 加上 transform 中新增的 variable，否则继承上一个组件的 variable。

案例

设置 payload

1. 添加 payload。

Transform ?

配置

备注

输入信息

payload any

attributes dict

- method string
- host string
- requestUri string
- requestPath string
- queryParams multimap
- uriParams multimap
- headers multimap
- cookies dict
 - remoteAddr string
- variables dict
 - company any

输出信息 +

添加转换属性

类型 payload

确认

取消

拖拽属性建立转换关系

2. 添加数据类型。

Transform ?

配置

备注

输入信息

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

company any

输出信息

payload x +

拖拽属性建立转换关系

any

string

int

float

bool

decimal

datetime

date

time

绑定自定义数据类型

添加自定义数据类型

绑定数据类型

编辑自定义数据类型

×

名称

Custom_Type

数据类型

Entity

请选择一种Entity类型

JSON

1

{

2

"name": "a",

3

"age": 10

4

}

预览

▼ Custom_Type <entity>

name string

age int

确定

取消

3. 绑定数据类型。

Transform ?

配置

备注

输入信息

payload any

attributes dict

- method string
- host string
- requestUri string
- requestPath string
- queryParams multimap
- uriParams multimap
- headers multimap

cookies dict

- remoteAddr string

variables dict

- age int

输出信息

payload × +

拖拽属性建立转换关系

any

string

int

float

bool

decimal

datetime

date

time

绑定自定义数据类型

添加自定义数据类型

绑定数据类型

版权所有：腾讯云计算（北京）有限责任公司

第142 共732页

4. 拖动字段，进行匹配。

Transform ?

配置

备注

输入信息

payload any

attributes dict

- method string
- host string
- requestUri string
- requestPath string
- queryParams multimap
- uriParams multimap
- headers multimap

cookies dict

- remoteAddr string

variables dict

- age int

输出信息

payload x +

Custom_Type <entity>

name string

age int

版权所有：腾讯云计算（北京）有限责任公司

第143 共732页

5. 若图形化映射无法满足业务需求，可使用脚本进行编辑。

Transform

配置

备注

输入信息

输出信息

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

age int

payload × +

Dataway表达式

process(msg):

{'name': msg.attrs['method'], 'age': msg.

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

attributes

attrs

vars

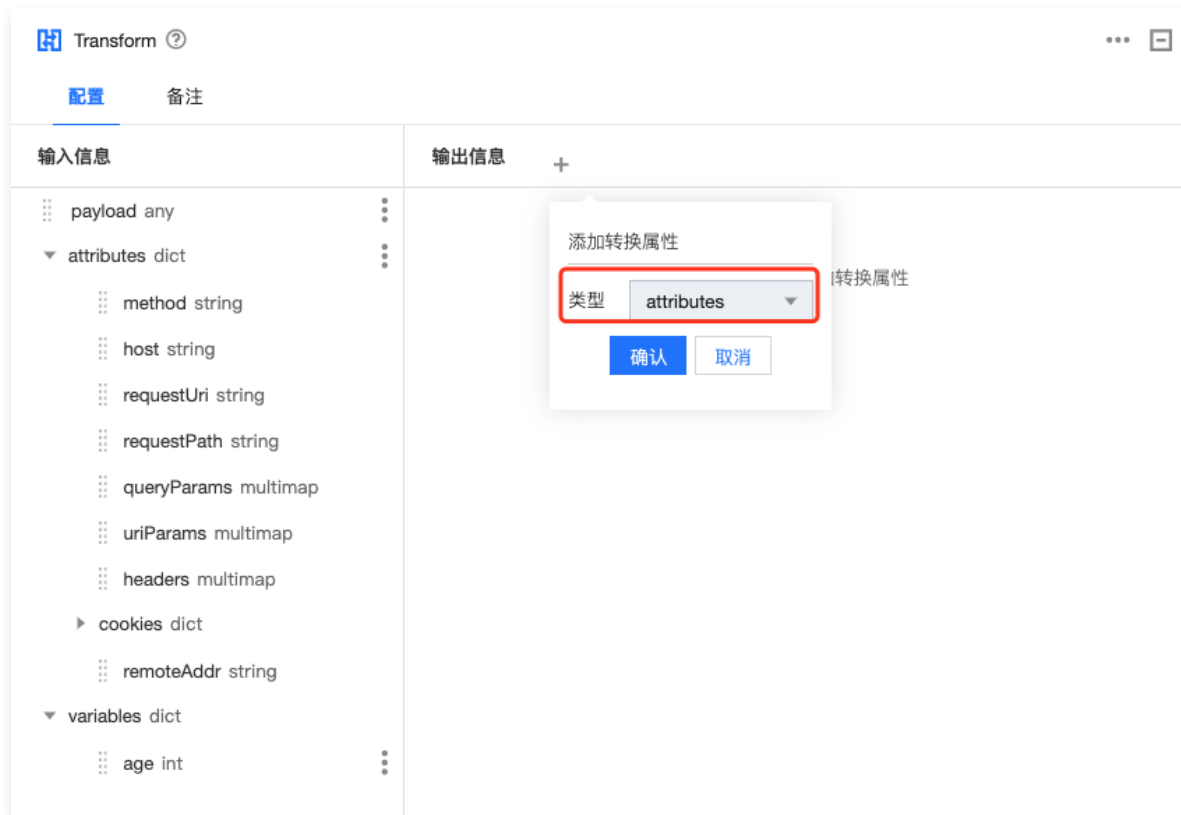
vars["age"]

图形转换

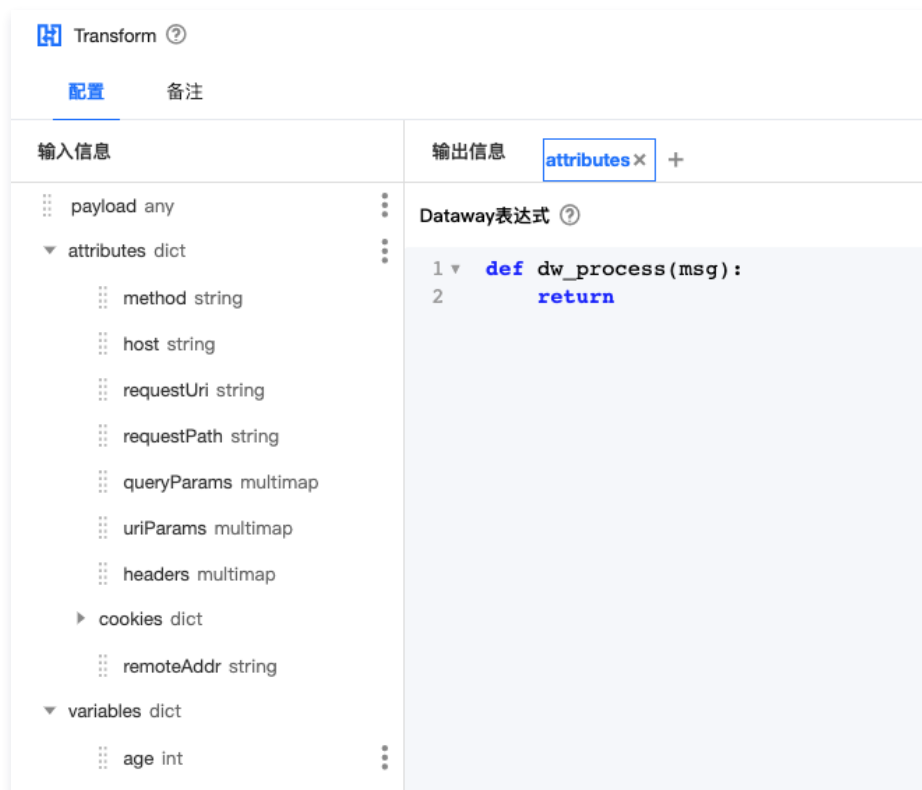
脚本输入

设置 attribute

1. 添加 attributes。



2. 添加表达式，编辑 attributes，attributes 的类型为 dict，因此表达式的输出需要保证类型为 dict，也可使用图形化编辑，使用方式与设置 payload 相同



设置 variable

1. 添加 variables，“名称”处填入要声明的变量名字，也可使用图形化编辑，使用方式与设置 payload 相同。

Transform ?

配置

备注

输入信息	输出信息 +
payload any attributes dict method string host string requestUri string requestPath string queryParams multimap uriParams multimap headers multimap cookies dict remoteAddr string variables dict age int	添加转换属性 类型 variables 名称 test 确认 取消

2. 添加表达式，编辑变量数据。

Transform ?

配置

备注

输入信息	输出信息 test x +
payload any attributes dict method string host string requestUri string requestPath string queryParams multimap uriParams multimap headers multimap cookies dict remoteAddr string variables dict age int	Dataway表达式 ? <pre> 1 def dw_process(msg): 2 return </pre>

Try

最近更新时间：2023-06-08 14:47:57

以下视频将为您介绍集成流开发时的错误处理：

[观看视频](#)

简介

Try 组件的作用是捕获错误，可以捕获 Try 中子流运行时抛出的错误和系统错误，也可与 Raise Error 组件搭配使用，捕获用户自定义错误，Raise Error 组件抛出错误后，Try 使用 Catch 捕获，并执行 Catch 中配置的子流。当 Catch 中配置的错误类型为"any"时，可以捕获所有错误。

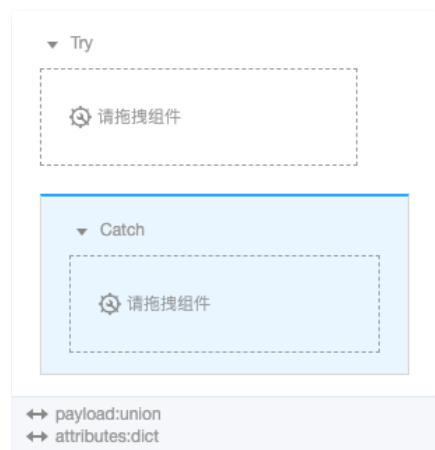
操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
错误类型	string	流执行过程中抛出的错误类型，配置"any"，可捕获所有错误。	是	无

界面配置

当 Try 中配置的子流抛出错误后，Catch 可根据配置的错误类型进行捕获，当 Catch 中配置的错误类型与 Raise Error 抛出的错误类型一致时，执行 Catch 中的补偿策略。



输入到子流中的 message

message 属性	值
payload	继承 Try 上一个组件的 payload。
error	空。
attribute	继承 Try 上一个组件的 attribute 信息。
variable	继承 Try 上一个组件的 variable 信息。

输出

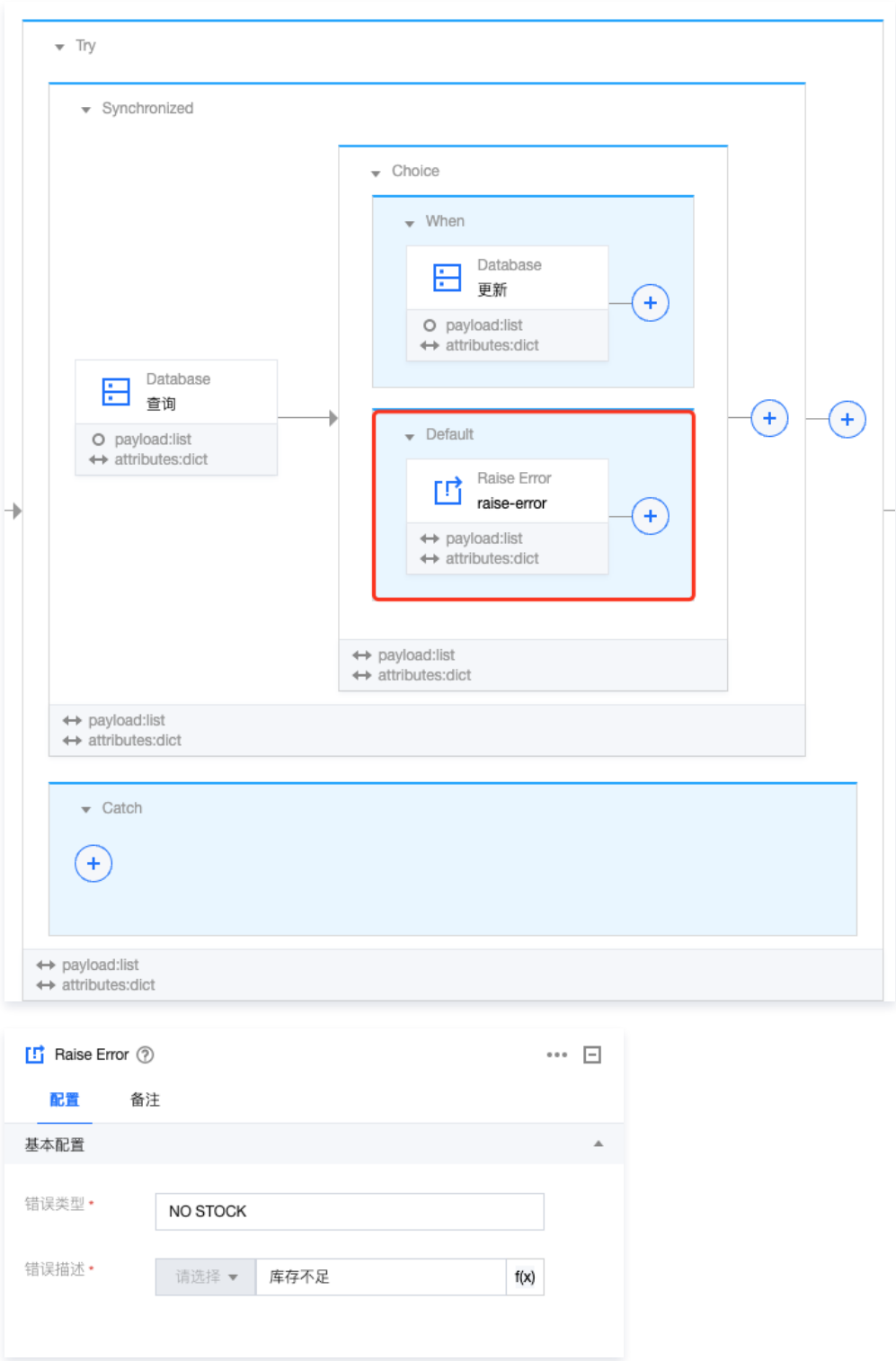
组件输出的 message 信息如下：

message 属性	值
payload	若 Try 中子流正常运行，payload 的结果为子流输出的 payload。若抛出错误，且错误被 Catch 捕获，则 payload 的结果为 Catch 中子流输出的 payload。若抛出的错误未被捕获，则流终止运行。
error	若抛出错误，error 保存错误信息，为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。

attribute	若 Try 中子流正常运行，attribute 的结果为子流输出的 attribute。若抛出错误，且错误被 Cache 捕获，则 attribute 的结果为 Cache 中子流输出的 attribute。若抛出的错误未被捕获，则流终止运行。
variable	若 Try 中子流正常运行，variable 的结果为子流输出的 variable。若抛出错误，且错误被 Cache 捕获，则 variable 的结果为 Cache 中子流输出的 variable。若抛出的错误未被捕获，则流终止运行。

案例

1. 当查询错误或返回值不满足条件时，抛出错误。



2. 使用 Try Cache 捕获错误。

Try ?

...

[-]

配置

备注

Catch ?

...

错误类型 *

NO STOCK

备注

请输入备注信息

Table

最近更新时间：2024-08-22 14:47:21

简介

Table 组件是一个由腾讯云数据连接器提供存储的内置表结构组件，提供基于Table的增删改查等操作，仅用于操作 RecordSet。
一般情况前一个节点为 RecordSet Encoder，将数据封装成 RecordSet，通过 Table 组件实现数据的查询、插入、合并、删除、比对等。使用 Table 组件时选择合适的操作 > 配置连接 > 配置操作。

连接配置

选择通用存储中创建的表名，此表为操作的目标表。

参数	数据类型	描述	是否必填	默认值
表名	string	表名	是	-

操作说明

Table 组件包含插入、查询、合并、删除、比对操作。

查询

参数配置

参数	数据类型	描述	是否必填	默认值
过滤条件	list	查询过滤条件	是	-
输出字段	list	选择要输出的字段	是	-
分区数	int	分数数量	是	-
开启缓存	bool	对输出 RecordSet 是否缓存	是	-

查询界面如下：

Table 查询

...

当前使用的是 tx_db_connect 连接配置

编辑 | 切换

配置

备注

查询逻辑配置

查询条件 *

序号	字段名	操作	目标值
1	code	==	string 01 f(x)

+添加变量

字段选择 *

☒ 字段名	字段类型	操作
☒ code	string	删除
☒ name	string	删除
☒ age	string	删除

输出配置

分区数

int 1 f(x)

开启缓存

bool false f(x)

高级配置

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为查询到的数据集构成的 RecordSet；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 新建连接器或选择已创建的连接器。
- 配置过滤条件、输出字段等信息。

配置

调试

备注

通用

过滤条件

序号	字段名	操作	目标值
1	姓名	==	string elena001 f(x)

+添加变量

输出字段

☒ 字段名	字段类型	操作
☒ 姓名	string	删除
☒ 年龄	int	删除

+添加变量

高级配置

分区数

int

1

f(x)

开启缓存

bool

true

f(x)

3. 执行操作，输出结果如下：

输出消息 payload

```
{
  "id": 66,
  "partitions": 1,
  "records": [
    [
      "elena001",
      15
    ]
  ],
  "schema": [
    "name(string)",
    "age(int)"
  ]
}
```

下载

插入

参数配置

参数	数据类型	描述	是否必填	默认值
忽略错误	bool	忽略插入错误	否	true
输入数据集	RecordSet 类型	输入数据集，若未填写，默认为 message 的 payload	是	-

字段校准	list	对输入数据集进行字段校准	否	-
字段映射	enum	字段映射，数据表字段和 RecordSet 类型数据字段的映射，插入数据表字段的值为 RecordSet 类型数据字段的值	是	-

默认插入界面如下：

Table 插入

当前使用的是 tx_db_connect 连接配置

配置

备注

输入配置

输入数据集

Record: def dw_process(msg): return msg

插入逻辑配置

字段映射

表字段	输入字段	操作
code	code	删除
name	name	删除
age	age	删除

+新增字段

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

- 说明
- 如果前一个节点输出 RecordSet，Table 将自动感知数据结构，并将来源字段与目标表字段根据同名映射自动进行映射。
 - 在字段映射后方有 schema 维护按钮，可单击查看、管理输入的 schema 在 schema 维护面板中可进行手工维护 schema 的字段信息，当前一节点输出的 schema 发生了调整，可在 schema 手工刷新与前一节点输出的 schema 保持一致，同时将根据同步映射原则进行重新映射（将覆盖上次的映射）。

输入数据集Schema校准

✕

🔄

 点击刷新后将根据感知的Schema进行更新，同时会更新映射关系

字段名	字段类型	操作
code	string	删除
name	string	删除
age	string	删除
+新增字段		

确定

取消

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为上一个组件的 payload；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 新建连接器或选择已创建的连接器。
2. 配置字段映射等信息。

Table ? 插入

...

当前使用的是 tx_db_connect 连接配置

编辑 | 切换

配置

备注

输入配置

▲

输入数据集

Record: ▼

def dw_process(msg): return msg

🗑️

插入逻辑配置

▲

字段映射 *

✎

表字段	输入字段	操作
code ▼	code ▼	删除
name ▼	name ▼	删除
age ▼	age ▼	删除

+新增字段

高级配置

▲

输出消息绑定自定义数据类型

消息属性

payload ▼

类型选择

空 ▼

+

3. 执行操作，如果成功，则输出 payload 为上一个组件的 payload。

合并

参数配置

参数	数据类型	描述	是否必填	默认值
只执行插入	bool	数据不存在时执行插入操作	是	false
只执行更新	bool	数据存在时执行更新操作	是	false
忽略错误	bool	忽略插入错误	否	true
输入数据集	RecordSet 类型	输入数据集，若未填写，默认为 message 的 payload	是	-
字段校准	list	对输入数据集进行字段校准	否	-
过滤条件	list	合并的过滤条件	是	-
字段映射	enum	字段映射，数据表字段和 RecordSet 类型数据字段的映射，插入数据表字段的值为 RecordSet 类型数据字段的值	是	-

Table 合并

当前使用的是 tx_db_connect 连接配置 [编辑](#) | [切换](#)

配置

备注

输入配置

输入数据集

Record: def dw_process(msg): return msg

合并逻辑配置

只执行插入

bool false

f(x)

只执行更新

bool false

f(x)

主键配置

表字段

code

输入字段

book_id

过滤条件

序号	字段名	操作	目标值
+添加变量			

字段映射

表字段	输入字段	是否更新	操作
code	请选择	是	删除
name	请选择	是	删除

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为上一个组件的 payload；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素： “Code” 表示错误类型， “Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 新建连接器或选择已创建的连接器。
2. 设置输入数据集，默认为return msg.payload。

3. 配置过滤条件，字段映射等信息。

通用

仅插入

bool

false

f(x)

仅更新

bool

false

f(x)

忽略错误

bool

true

f(x)

输入配置

输入数据集

输入数据集

Record

def dw_process(msg): return ms

输入字段

字段名	字段类型	操作
+添加变量		

合并配置

过滤条件

序号	字段名	操作	目标值
1	姓名	==	name1
+添加变量			

字段映射

表字段	输入字段	是否更新	操作
年龄	age1	是	删除
姓名	name1	是	删除
+添加变量			

字段映射 *

表字段	输入字段	是否更新	操作
code	book_id	是	删除
name	book_name	是	删除
age	book_auth	是	删除
+新增字段			

4. 执行操作，如果成功，则输出 payload 为上一个组件的 payload。

删除

参数配置

参数	数据类型	描述	是否必填	默认值
输入数据集	RecordSet 类型	输入数据集，若未填写，默认为 message 的 payload	是	-
字段校准	list	对输入数据集进行字段校准	否	-
过滤条件	list	删除的过滤条件	是	-

Table ? 删除

...

当前使用的是 tx_db_connect 连接配置

编辑 | 切换

配置

备注

输入配置

输入数据集

Record: def dw_process(msg): return ms

删除图标

删除逻辑配置

主键配置

主键配置图标

表字段

code

输入字段

请选择

过滤条件 *

序号

字段名

操作

目标值

+添加变量

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为上一个组件的 payload；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为dict类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 新建连接器或选择已创建的连接器。

2. 配置删除逻辑等信息。

删除逻辑配置

主键配置

表字段

code

输入字段

book_id

过滤条件 *

序号	字段名	操作	目标值
1	code	In	book_id

+添加变量

3. 执行操作，如果成功，则输出 payload 为上一个组件的payload。

比对

参数配置

参数	数据类型	描述	是否必填	默认值
输入数据集	RecordSet 类型	输入数据集，若未填写，默认为 message 的 payload	是	-
字段校准	list	对输入数据集进行字段校准	否	-
输入模式	enum	全量、增量	是	-
主键配置	list	计算时主键的映射关系	是	-
比对字段	list	计算时需要根据主键是否相等	是	-
开启缓存	bool	对输出 RecordSet 是否缓存	是	-

Table 比对

...

当前使用的是 tx_db_connect 连接配置

编辑 | 切换

配置

备注

输入配置

▲

输入数据集

Record:

def dw_process(msg): return ms

🗑

比对逻辑配置

▲

输入模式 *

全量模式

主键配置 *

✎

表字段

输入字段

code

请选择

比对字段 *

✎

表字段

输入字段

操作

code

请选择

删除

name

请选择

删除

age

请选择

删除

+新增字段

输出配置

▲

分区数

int

1

f(x)

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为计算差值后生成的 RecordSet；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

输出 RecordSet 中 op_type 字段取值为 Add、Modify、Deleted，分别表示新增、修改、删除。

案例

1. 新建连接器或选择已创建的连接器。

2. 配置输入模式、主键配置、比对字段等信息。

比对逻辑配置

输入模式 *

全量模式

主键配置 *

表字段

输入字段

code

请选择

比对字段 *

表字段	输入字段	操作
code	请选择	删除
name	请选择	删除
age	请选择	删除

+新增字段

3. 执行操作，输出如下：

输出消息 payload

```
"id": 44,
"partitions": 1,
"records": [
  [
    "Modify",
    15,
    "\"elena001\"",
    14,
    "\"elena001\""
  ],
  [
    "Modify",
    80,
    "\"zxl002\"",
```

下载

Until Successful

最近更新时间：2023-06-08 14:47:57

简介

Until Successful组件中可以配置子流，该组件的作用是对子流执行重试操作。Until Successful中可支持3种重试类型：无条件重试、按成功条件重试、按失败条件重试。用户可根据需要配置重试次数，支持的范围为1-100。两次重试之间的时间间隔也可根据需求配置，当前支持的范围为1 - 300秒。

- 无条件重试：如果重试条件类型为无，则只在子流报错的情况下重试。
- 按成功条件重试：当表达式结果为 false 时重试。
- 按失败条件重试：当表达式结果为 true 时重试。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值
重试条件类型	无、按成功条件重试、按失败条件重试	重试类型，按成功条件重试和按失败条件重试需填入条件表达式	是	无
重试次数	int	重试次数，取值范围：1 - 100	是	3
重试时间间隔	int	重试间隔，取值范围：1 - 300，单位：秒	是	60

配置界面

Until Succe... ?

配置 输入 输出 备注

...

基本配置

重试条件类型 *

按成功条件重试

重试条件 *

def dw_process(msg): return m

设置重试条件

重试次数 *

3

最大重试次数,取值范围[1,100]

重试时间间隔 *

60

两次重试的最小时间间隔,取值范围[1,300],单位秒

输入到子流中的message

message 属性	值
payload	继承 Until Successful 上一个组件的 payload
error	空
attribute	继承 Until Successful 上一个组件的 attribute 信息
variable	继承 Until Successful 上一个组件的 variable 信息

输出

组件输出的 message 信息如下：

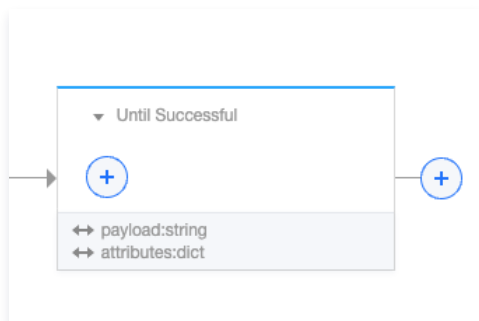
message 属性	值
------------	---

payload	继承子流输出的 payload
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承子流的 attribute 信息
variable	继承子流的 variable 信息

案例

当需要对某类操作进行重试时，可以使用该组件，例如发送 request 请求，当请求失败时，使用 Until Successful 进行重试。

1. 添加 Until Successful 组件。



2. 选择按成功条件重试，当服务器返回的 errcode 不等于0时，我们重新发送请求，设置重试次数为3，重试间隔为1秒。

Until Succe... ?

配置

备注

基本配置

重试条件类型 *

按成功条件重试

重试条件 *

def dw_process(msg): return msg.payload.get('errcode') != 0

重试次数 *

int 3

重试时间间隔 *

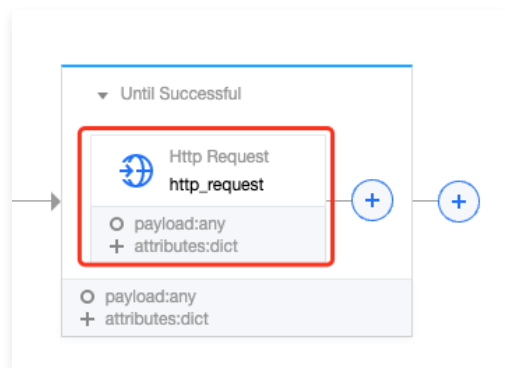
int 1

重试条件如下：

Dataway表达式 ?

```
1 def dw_process(msg):
2     return msg.payload.get('errcode') == 0
```

3. 在 Until Successful 组件中添加 Request 组件，用于发送 HTTP 请求。



While

最近更新时间：2023-06-08 14:47:57

简介

While 组件是循环类组件，While 组件支持的最大循环次数为10000。与 For Each 组件不同，While 组件的循环并不依赖数据集，而是使用条件语句判断循环是否需要继续，当循环条件为 True 时，会继续循环，直到返回 False 或达到最大的循环次数。我们将输入到 While 组件的 message 消息称作父 message，While 组件中配置的子流，会生成子 message，子 message 只继承父 message 的 variable 变量，并且父 message 的消息内容，全部保存在变量 rootMessage 中，如果要在子流中访问父 message 的 payload，使用表达式 msg.vars.get('rootMessage').payload。当 While 组件执行完成后，只有子流中的 variable 变量可以传递出来，并添加到父 message 中，继续向下传递。

操作配置

参数配置

参数	数据类型	描述	是否必须	默认值
循环条件	bool	返回 True，则继续执行循环；返回 False，终止循环。	是	无
计数器	string	计数器是一个变量，该变量负责存储当前的迭代次数，从0开始，此处填入变量名称，msg.vars.get('#计数器变量#') 即可使用；例如：当计数器变量使用默认值 counter 时，第1次循环，msg.vars.get('counter') 值为0，第2次循环，msg.vars.get('counter') 值为1。	是	count er
最大循环次数	int	最大循环次数，超过后终止循环，最大值为10000。	是	10000

配置界面

While ?

配置 输入 输出 备注

基本配置

循环条件

点击输入DataWay表达式

执行循环的条件,返回true继续循环,false结束循环

计数器

counter

循环遍历的计数器

最大循环次数

10000

循环的最大次数,超过后终止循环

输入到子流中的 message

message 属性	值
payload	空。
error	空。
attribute	空。
variable	继承主流中的 variable 数据，同时新增两个变量，分别为计数器和根信息，若用户使用默认值，可使用表达式 msg.vars.get('counter') 和 msg.vars.get('rootMessage') 进行访问。

输出

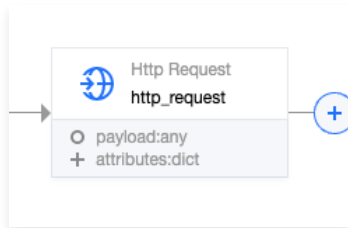
组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload。
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	上个组件的 variable 信息加上 While 中子流新增的 variable 变量。

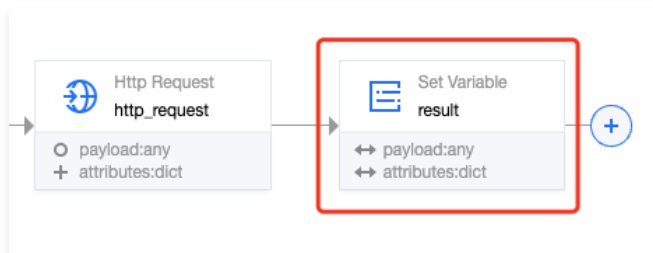
案例

使用 While 组件，我们可以执行一些数据的筛选，例如：从服务端返回的数据列表中筛选符合业务需求的数据。

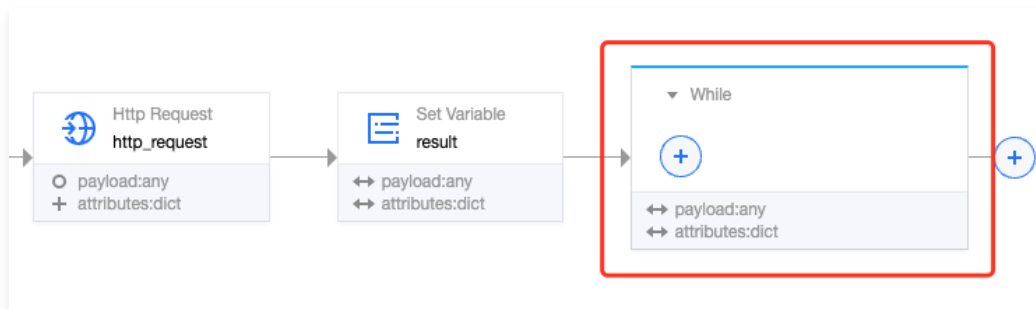
1. 使用 Request 组件请求服务端数据。



2. 声明保存结果的变量。



3. 添加 While 组件，循环结束的条件是 “result” 变量不为空，计数器使用默认 “counter”。



While ?

...

[-]

配置 备注

基本配置

循环条件 *

def dw_process(msg): return msg.vars.get('res')

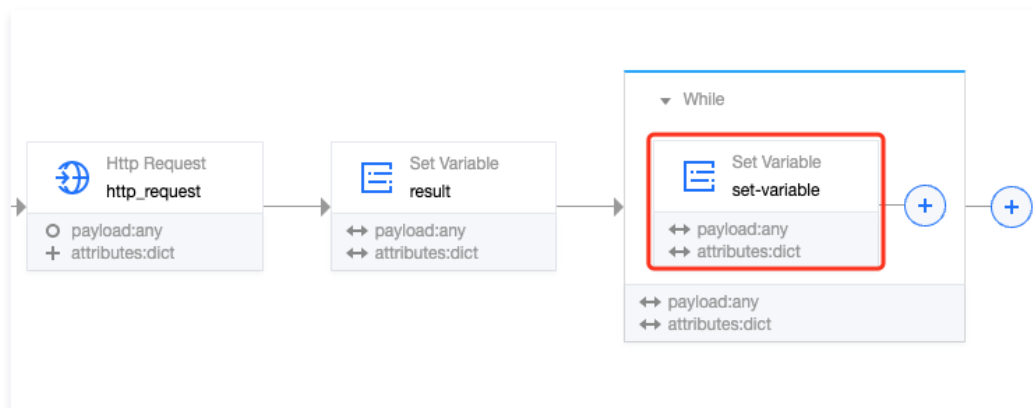
计数器 *

counter

最大循环次数 *

请选择 10000

4. 添加 variable 变量，筛选数据。



Set Variable ?

配置 备注

基本配置

变量名 *

变量值 * 请选择 ▼ def dw_process(msg): return 🗑️

5. index 变量获得数组下标，payload 中的 data 为数组类型，根据下标获取数据，判断数据是否符合要求。若符合，则返回结果。While 组件的下个节点可使用 msg.vars.get('result') 获得 While 子流中筛选到的数据。

Set Variable ?

配置 备注

配置 / 编辑

Dataway表达式 ?

```
1 def dw_process(msg):  
2     index = msg.vars.get('counter')  
3     data = msg.payload.get('data')[index]  
4     if data.get('name') == 'test'  
5         return data.get('city')  
6     return ''
```

Compress

最近更新时间：2023-06-08 14:47:57

简介

Compress 组件可对数据进行压缩和解压，支持的压缩和解压方式有：DEFLATE、GZIP、ZLIB。

操作说明

压缩操作

参数配置

参数	数据类型	描述	是否必填	默认值
待压缩内容	binary	待执行压缩操作的内容	是	-
压缩方式	enum	压缩方式，支持DEFLATE、GZIP、ZLIB	是	-
压缩等级	enum	压缩等级，支持默认压缩、完全不压缩、自定义压缩	是	默认压缩
自定义压缩等级	int	用户可自定义压缩等级，等级范围为[1,9]	否	-

配置

备注

通用

待压缩内容 *

请输入表达式

🗑

压缩方式 *

请选择

▼

f(x)

压缩等级

默认压缩

▼

f(x)

输出

查询操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回压缩后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 添加 Compress 连接器组件，选择压缩操作。
- 在通用配置中，填写相关参数。例如，待压缩内容使用 Dataway 表达式输入：

```
def dw_process(msg):
```

```
return Entity.from_bytes('ipaas-test', mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

配置

调试

备注

通用

待压缩内容

def dw_process(msg): return Entity.from_bytes

压缩方式

GZIP

压缩等级

自定义压缩

自定义压缩等级

int

5

等级范围为1-9，数值越大，压缩速度越慢，压缩效果越好

3. 执行成功后，message payload 中包含压缩后的二进制内容：

输出消息 payload

[
31,
139,
8,
0,
0,
0,
0,
0,
0,
255,
202,
44,
72,
76,
44,
214,
45,
73,
45,
46,
1,
4,
0,
0,
255,
255,
167,
86,
135,
66,
10,
0,
0,
0
]

解压操作

参数配置

参数	数据类型	描述	是否必填	默认值
待解压内容	binary	待执行解压操作的内容	是	-

解压方式	enum	解压方式，支持 DEFLATE、GZIP、ZLIB	是	-
------	------	---------------------------	---	---

配置

备注

通用

待解压内容 *

请输入表达式

解压方式 *

请选择

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 binary 类型，返回解压后的内容；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Compress 连接器组件，选择解压操作。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：待解压内容使用 Dataway 表达式输入：

```
def dw_process(msg):  
    return Entity.from_bytes(msg.payload, mime_type='text/plain', encoding='utf-8')
```

配置界面如下：

配置

调试

备注

通用

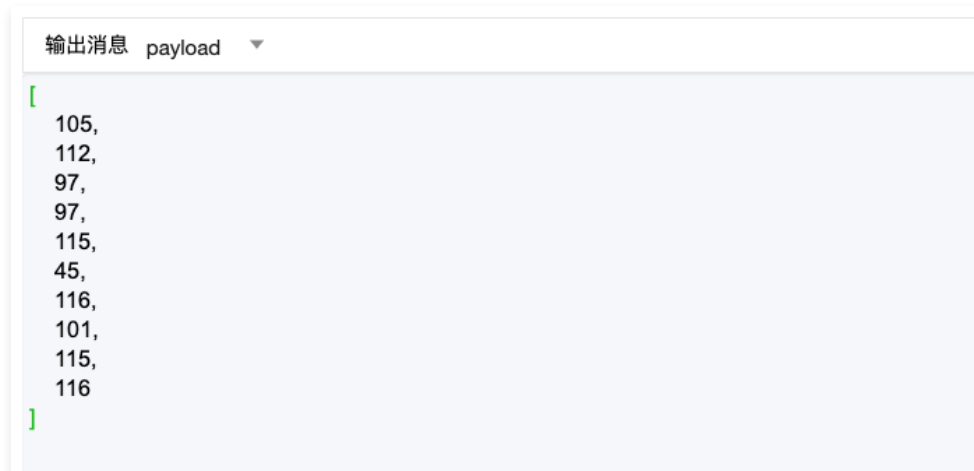
待解压内容 *

def dw_process(msg): return Entity.from_bytes

解压方式 *

GZIP

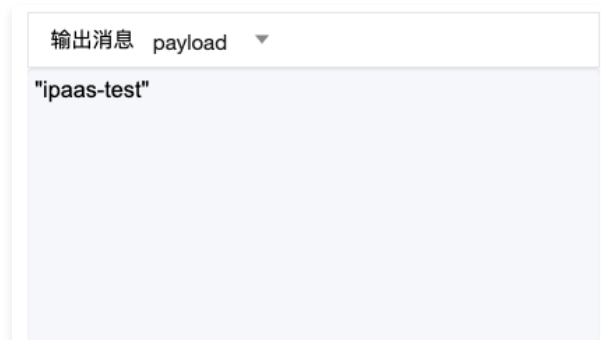
4. 插入成功后，message payload 中包含执行结果：



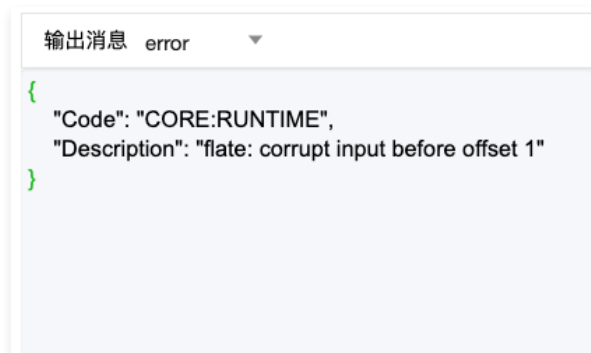
- 该结果可读性较差，可以对二进制内容进行解码，例如：添加“Set Payload”组件，输入 Dataway 表达式：

```
def dw_process(msg):  
    return msg.payload.decode('utf-8')
```

- “Set Payload” 组件的输出如下，解压后的值和压缩前的内容一致：



5. 若插入过程中出现错误，message error 中会包含错误信息：



Rate Limiter

最近更新时间：2023-06-08 14:47:57

简介

Rate Limiter 属于流控频控类组件，其作用是限制单位时间内（秒、分钟）通过 Rate Limiter 的总次数，从而达到控制流量大小的效果。Rate Limiter 使用的算法为令牌桶算法，会以一定速度匀速地补充令牌（如：限制每分钟执行10次，则每6秒补充一个令牌）。Rate Limiter 允许在有令牌积累的情况下，短时间内峰值流量达到设置的最大处理次数的两倍。

操作配置

参数配置

参数	数据类型	描述	是否必填	默认值	备注
时间单位	下拉选择	限流的时间单位，可选“秒”、“分”	是	无	-
最大处理次数	int	单位时间内允许的最大处理次数	是	无	在有令牌积累的情况下，短时间内允许的峰值流量为最大处理次数的两倍。
限流策略	单选框	可选择“拒绝处理”或者“在xx秒内处理超限部分”	是	拒绝处理	“拒绝处理”：获取不到令牌直接报错。“在xx秒内处理超限部分”：获取不到令牌时，根据获取令牌所需要的时间进行排队等待，达到削峰填谷的效果。

界面配置

限流 ?

...

配置备注

基本配置

限制每秒最大处理100次

超过最大次数后：
☒ 拒绝处理
☐ 在最大值10秒内处理超限部分

输入 message

message 属性	值
payload	继承 Rate Limiter 上一个组件的 payload 信息。
error	继承 Rate Limiter 上一个组件的 error 信息。
attribute	继承 Rate Limiter 上一个组件的 attribute 信息。
variable	继承 Rate Limiter 上一个组件的 variable 信息。

输出 message

message 属性	值
payload	若能够获取到令牌，payload 的结果为继承 Rate Limiter 上一个组件的 payload。

error	“直接拒绝”策略下，获取不到令牌，错误信息为”获取不到令牌，触发限流“。”在xx秒内处理超限部分“策略下，超过最大等待时间，错误信息为”获取令牌所需要的时间超过最大等待时间，触发限流“
attribute	若能够获取到令牌，attribute 的结果为继承 Rate Limiter 上一个组件的 attribute。
variable	若能够获取到令牌，variable 的结果为继承 Rate Limiter 上一个组件的 variable。

配置案例

拒绝处理

- 配置：限制每分最大处理2次。选择**拒绝处理**。

限流 ?

...

配置

备注

基本配置

限制每 分 最大处理 2 次

超过最大次数后:

☒ 拒绝处理

☐ 在 1 分内处理超限部分

- 若获取不到令牌。

输出消息 error

复制

错误类型

系统错误:内部错误

错误描述

内部错误:获取不到令牌，触发限流

在xx秒内处理超限部分

- 配置：限制每分最大处理2次。选择：“在2分内处理超限部分”。

限流 ?

...

配置

备注

基本配置

限制每 分 最大处理 2 次

超过最大次数后:

☐ 拒绝处理

☒ 在 2 分内处理超限部分

- 若获取不到令牌，并且超过最大等待时间。

输出消息

error

▼

复制

错误类型

系统错误:内部错误

错误描述

内部错误:获取令牌所需要的时间超过最大等待时间，触发限流

API 中心

API 管理

最近更新时间：2024-11-06 15:22:02

 注意

API 管理的独立环境与应用集成环境不互通。

各大企业每天都有大量的 API 增长，同时越来越多公司开始公开 Web API，API 的使用场景正在累积。现在，每日 API 调用量在不断飙升，如何能够安全有效将这些 API 管理起来对于企业而言并不容易。

腾讯云数据连接器提供 API 发布功能，可以一键将已发布的应用打包生成 API，方便用户进行管理和调用；同时提供了 API 管理能力，可以针对 API 进行访问权限管控和流量调度。

API 管理主页

登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击 **API 中心 > API 管理**，即可进入 API 管理的主页。

在 API 管理主页，您可以创建或查看 API，API 列表中展示有 API 服务名称、API 服务状态、API 服务域名、API 更新时间、API 服务的数据统计、endpoint 统计、API 服务的鉴权方式以及操作等内容。同时，每个 API 服务均可展开，查看下属的 endpoint 的信息和状态。

API服务管理

默认项目

帮助文档

创建API服务

请输入API服务名称

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test2 v1.0.1	开发中	0	http://...-137.api.pa...	2021-12-30 15:22:41		1		0	编辑 添加endpoint 更多操作
API_testYT v1.0.1	上线	0	https://...-155.api.pa...	2021-12-30 15:22:41		1	BasicAuth/OAuth2.0	2	查看 添加endpoint 更多操作
API_test11 1.0.0	下线	AA的标签	http://...-770.api.pa... https://...-770.api.pa...	2021-12-30 15:22:41		1	BasicAuth	0	查看 添加endpoint 更多操作

访问路径

调用方法

分组

路径地址

后端服务

最近修改时间

操作

/getUserInfo	GET		http://...-770.api.pa... https://...-770.api.pa...	第三方服务/http://baidu.com	2021-12-29 19:2...	查看 删除
--------------	-----	--	---	------------------------	--------------------	-------

共 1 条

10 条 / 页

1 / 1 页

API_test1 1.0.5	开发中		http://...-797.api.pa...	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作
-----------------	-----	--	--------------------------	---------------------	--	---	--	---	--------------------

共 4 条

10 条 / 页

1 / 1 页

创建 API 服务

API 管理功能支持 3.0.0 版本的 OpenAPI 规范。OpenAPI 3.0.0 规范的对象定义请参见 [OpenAPI Specification](#)。用户可以通过单击**创建 API 服务**进入 API 创建界面。

创建 API 服务可以通过 API 描述文件创建，也可以通过页面手动创建。

通过 API 描述文件创建

1. 在 **API 管理** 页面，单击**创建 API 服务**，在基本配置中配置以下信息，完成后单击**下一步：策略信息**。

创建方式：选择**创建自配置文件**。

上传描述文件：上传 YAML 或 JSON 格式的文件，大小100KB以内。

支持格式：YAML、JSON。

←

创建API服务

1 基本配置

>

2 策略信息

创建方式

● 创建自配置文件

○ 手动创建

上传描述文件

case.yaml

重新上传

删除

请上传 yaml/json 格式文件, 大小 100KB 以内

文件格式

YAML

格式校验

```

1 openapi: 3.0.0
2 info:
3   title: Simple API overview
4   version: 2.0.0
5   description: test desc
6 paths:
7   /:
8     get:
9       description: get desc
10      operationId: listVersionsv2
11      summary: List API versions
12      responses:
13        '200':
14          description: 200 response
15          content:
16            application/json:
17              examples:
18                foo:
19                  value:
20                    versions:
21                      - status: CURRENT
22                        updated: '2011-01-21T11:33:21Z'
23                        id: v2.0
24                        links:
25                          - href: http://127.0.0.1:8774/v2/
26                            rel: self
27                      - status: EXPERIMENTAL
28                        updated: '2013-07-23T11:33:21Z'
29                        id: v3.0
30                        links:

```

2. 若解析成功, 单击完成, 则可以直接通过此文件生成对应的 API 服务。

通过页面手动创建

- 在 [API 管理](#) 页面, 单击 **创建 API 服务**, 在基本配置中配置以下信息, 完成后单击下一步: **策略信息**。
 - 创建方式: 选择“手动创建”。
 - API 名称: 请输入 API 名称。
 - 协议: 支持 HTTP、HTTPS、HTTP 和 HTTPS 协议类型。
 - 版本: 请输入版本号。
 - 描述 (选填): 对此 API 服务的一些简单描述。
 - 标签 (选填): 如果有分组的需要, 希望日后可以通过标签来进行快速的 API 服务筛选工作, 还可以配置标签信息。

← 创建API服务

1 基本配置



2 策略信息

创建方式 *



创建自配置文件



手动创建

API名称 *

请输入API名称

请输入API名称

协议 *

HTTP

HTTPS

HTTP&HTTPS

版本 *

请输入版本

请输入版本号

描述

请输入描述

标签 (选填)

请输入标签名

上一步

下一步: 策略信息

2. 在策略信息中配置以下信息，单击完成即可创建一个 API 服务。

- 配置鉴权策略：支持 BasicAuth、OAuth2.0 和无验证（即不填写检测策略）。
- 配置黑白名单：您可以按需开启，开启后可以输入多条 IP 进行黑白名单的访问限制。
- 开启访问限制：
 - 请求频率策略：从配置时间起，每单位时间内允许的最大请求次数，填写范围为1 - 1000。
 - 访问限制策略：从配置时间算起，每单位自然时间内允许的最大访问次数，填写范围为1 - 1000。

←

创建API服务

✓

基本配置

2

策略信息

配置鉴权策略

请选择鉴权策略

配置黑白名单

配置白名单

请求IP白名单

多个IP以换行分隔

开启访问限制

请求频率策略

—

100

+

次/

日

▼

请输入请求频率

访问限制策略

—

100

+

次/

日

▼

请输入访问限制

上一步

完成

创建 API Endpoint

当我们创建好一个 API 服务之后，则代表此 API 已经生成，可以开始编辑其具体行为。

API Endpoint 指一个 API 服务可以连接的后端服务信息，包括 API 的访问路径（API Path）、API 的调用方式、分组、API Endpoint 绑定的后端服务类型，最近修改时间以及一些操作。

访问路径	调用方法	分组	路径地址	后端服务	最近修改时间	操作
/v2	GET		http://-915.api.ipa...		2022-01-05 14:2...	编辑 删除 调试
/	GET		http://-915.api.ipa...		2022-01-05 14:2...	编辑 删除 调试

共 2 条

10 条 / 页

1

/ 1 页

默认新创建好的 API 服务是没有已绑定的 Endpoint 信息的，需要进行创建完成首次的绑定工作。

- 在 [API 管理](#) 页面，单击目标 API 服务名称，进入服务详情页。
- 单击添加 API Endpoint，进行 API Endpoint 基本配置，完成后单击下一步：策略信息。
 - 访问路径：访问路径必须要以 “/” 开头。
 - 后端服务：支持选择现成的集成流或者绑定第三方服务 BASE URL。
 - 当选择的后端服务为“集成流”模式时，集成流只可以选择已发布的，并且配置了拥有 HTTP 监听能力组件的流。
 - 当选择“第三方服务 BASE URL”时，在资源路径处输入资源链接。
 - 描述：选填。

←

创建API

1 基本配置

>

2 策略信息

>

3 参数信息

访问路径 *

/

请输入API的访问路径,例: /getUserInfo

后端服务 *

集成流

第三方服务BASE URL

集成流 *

应用名称

集成流名称

发布环境

请选择集成流

描述

请输入描述

上一步

下一步: 策略信息

3. 在策略信息页面，配置以下信息，完成后单击下一步：参数信息。

① 说明：

- 与 API 服务的策略信息配置方式类似，允许用户在此 Endpoint 所归属的 API 服务的策略限制基础上，配置第二层策略限制。两层限制共同生效。
- 配置鉴权策略：您可以单击“同步服务策略”，一键同步上层的 API 服务策略，或者手动填写新的管理策略。
此处可选择的鉴权策略范围为上层 API 服务的鉴权策略，即若上层 API 服务只允许 OAuth 2.0 的鉴权方式，则下层的 API Endpoint 只允许配置 OAuth 2.0，而不能配置 Basic Auth。

←

编辑API

✓ 基本配置

>

2 策略信息

>

3 参数信息

配置鉴权策略

请选择鉴权策略

同步服务策略

配置黑白名单

配置黑白名单

请求IP白名单

多个IP以换行分隔

开启访问限制

请求频率策略

- 100 + 次/ 日

请输入请求频率

访问限制策略

- 100 + 次/ 日

请输入访问限制

上一步

下一步: 参数信息

4. 在参数信息页面，您可以配置访问此 API Endpoint 时可以选用的方法（API Method）和每种方法对应的调用参数信息。

- API 调用方法可以多选，若想取消，则可以反选想要删除的方法。
- 每一个参数后都提供了一系列操作，分别为：向上移动、向下移动、删除此参数，为此参数添加备注以及选择该参数是否必填。

编辑API

基本配置

策略信息

3 参数信息

GET POST PUT DELETE HEAD

URL Params Http Headers Response

属性名

类型

设为必填

操作

item

string

☒

↑

↓

🗑️

📄

+ 添加

上一步 完成

5. 当上述配置全部完成后，单击完成，则会返回 API 列表，同时已经创建好的 Endpoint 信息将会展现在此。

API 测试与运维

腾讯云数据连接器 API 管理平台同时还提供了完整的运维方案。查看步骤如下：

1. 在 [API 管理](#) 页面，找到目标 API 服务，单击操作列的数据统计图标，进入运维页面。

API服务管理 默认项目 帮助文档

创建API服务 请输入API服务名称

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test1aaa 1.2.1	开发中		http://100000076228-914.api.lpa...	2022-01-05 14:25:08		0		0	编辑 添加endpoint 更多操作
API_test2 1.0.1	开发中	0	http://100000076228-137.api.lpa...	2021-12-30 15:22:41		1		0	编辑 添加endpoint 更多操作
API_testYT 1.0.1	上线	0	https://100000076228-155.api.lpa...	2021-12-30 15:22:41		1	BasicAuth/OAuth2.0	2	查看 添加endpoint 更多操作
API_test11 1.0.0	下线	AA的标签	http://100000076228-770.api.lpa... https://100000076228-770.api.lpa...	2021-12-30 15:22:41		1	BasicAuth	0	查看 添加endpoint 更多操作
API_test1 1.0.5	开发中		http://100000076228-797.api.lpa...	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作

共 5 条 10 条 / 页 1 / 1 页

2. 在运维页面，您可以查看此 API 在规定时间范围内的请求记录和状态汇总。

API日志 默认项目

日志 统计

API_testYT v1.0.1 全部路径

请输入用户ID 请输入作业标识 请选择日志级别 近1天

搜索

日志数量 0条 下载日志

域名	请求路径	请求方法	用户ID	请求耗时	日志级别	作业标识	时间	日志内容	事件标识	payload
当前列表为空										

已加载完所有内容

同时，为了方便用户对配置完成的 API 进行在线的调试，我们还提供了通过控制台实时发送测试请求并且获取模拟测试结果的机制。操作步骤如下：

1. 在 [API 管理](#) 页面，单击目标 API 服务名称前面的 ▶，显示访问路径。

2. 在访问路径中，单击操作列的调试，进入 API 调试页面。

API服务管理

默认项目

帮助文档

创建API服务

请输入API服务名称

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test1aaa	1.2.1	开发中	http://100000076228-914.api.paas...	2022-01-05 14:25:08		0		0	编辑 添加endpoint 更多操作
API_test2	v1.0.1	开发中	0	http://100000076228-137.api.paas...	2021-12-30 15:22:41		1	0	编辑 添加endpoint 更多操作
API_testYT	v1.0.1	上线	0	https://100000076228-155.api.paas...	2021-12-30 15:22:41		1	2	查看 添加endpoint 更多操作

访问路径

调用方法

分组

路径地址

后端服务

最近修改时间

操作

test	POST	0	https://100000076228-155.api.paas...	集成流/HelloWorldtest/HelloWorld_Flow/天津	2021-11-18 11:3...	查看 删除 测试
------	------	---	--------------------------------------	---------------------------------------	--------------------	----------

共 1 条

10 条 / 页

1 / 1 页

API_test11	1.0.0	下线	AA的标签	http://100000076228-770.api.paas...	2021-12-30 15:22:41		1	BasicAuth	0	查看 添加endpoint 更多操作
API_test1	1.0.5	开发中		http://100000076228-797.api.paas...	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作

共 5 条

10 条 / 页

1 / 1 页

3. 在 API 调试页面，您可以配置此 API Endpoint 的请求 Header 和 Body 内容，并单击发送请求。

API调试

API信息

路径

https://100000076228-155.mock.api.paas.beta.myqcloud.com/test

请求协议

HTTPS

请求方法

POST

请求参数

URL Params

Http Headers

Request Body

json

raw

URL Params: {}

Content-Type

application/json

描述

无

测试结果

发送请求

暂无数据

随后即可获取到测试的结果。我们会将后端服务返回的 Response 状态码和结果返回给用户，方便进行进一步的调试工作。

测试结果

发送请求

200 OK

Response Headers:

Date: "Wed, 05 Jan 2022 10:18:14 GMT"

Content-Type: "application/json"

Content-Length: "106"

X-Ipaas-Api-Traceid: "f645f97c...a0897a"

Response Body:

Data: {}

ErrorCode: "OK"

Message: "Mock Response."

RequestId: "f645f97c...7a"

API 用户中心

最近更新时间：2024-11-06 15:22:02

腾讯云数据连接器会根据用户已经配置完成的 API 服务（包括其 Endpoint）生成用户可以调用的开放 API。在 API 用户中心中，我们将会帮助您更好的管理每个 API 服务的用户，以及分配用户对于 API 的使用权限。
[API 用户中心](#) 首页是以列表的形式展示当前可以配置的所有 API 服务信息，但是展示维度与“API管理”略有不同。

API用户中心

默认项目

审批管理

用户管理

请输入关键词搜索

Q

服务名称	版本	鉴权方式	用户数	最近更新时间	操作
API_test1	v1.0.1	BasicAuth/OAuth2.0	0	2021-11-15 15:18:00	查看

共 1 条

10 条 / 页

1

/ 1 页

用户管理

用户管理功能允许 API 的提供者帮助用户添加管理账号、分配权限，并获取请求 API 时所需要的所有必要信息。
API 服务提供方将会以类似管理者的视角去统一管理用户，允许查看用户的 OAuth 信息并分享给用户以便用户调用，或者进行用户的封禁和启用。

添加

请输入关键词搜索

Q

成员名称	状态	账号ID	用户邮箱	用户电话	操作
asd	审批通过	3			OAuth信息 禁用 启用 查看详细信息
dsa	审批通过	7			OAuth信息 禁用 启用 查看详细信息
ddd	审批通过	8			OAuth信息 禁用 启用 查看详细信息

共 3 条

10 条 / 页

1

/ 1 页

添加用户的时候需要提供以下用户信息，以便能够更好的记录和维护。同时，此处的用户名称和用户密码也将作为对应 API 服务的 Basic Auth 验证方式。

添加用户

基本信息

用户名称 *

服务名称 *

用户密码 *

联系邮箱 *

联系电话 *

所属企业

策略信息

开启访问限制

请求频率策略

-

100

+

次/

日

请输入请求频率

访问限制策略

-

100

+

次/

日

请输入访问限制

确认

关闭

当创建好用户之后，则可以通过[查看详细信息](#)按钮获取到用户的 client ID、client Secret 等内容。

用户详情

×

基本信息

用户名称 *

yetong

服务名称 *

API_testYT ▾

用户密码 *

👁

联系邮箱 *

联系电话 *

所属企业

验证信息

Client ID

👁

Client Secret

👁

策略信息

开启访问限制

☒

请求频率策略

—

100

+

次/

日 ▾

请输入请求频率

访问限制策略

—

100

+

次/

日 ▾

请输入访问限制

🔄

📄

☰

说明：

当前仅支持 API 服务提供方手动帮助客户添加成员信息和账户信息，客户的自助注册的能力暂不支持。

使用流程

步骤1：创建 API 服务及其 API Endpoint

具体操作请参见 [创建 API 服务](#)、[创建 API Endpoint](#)。

步骤2：获取 API 请求地址

1. 在 API 服务列表获取 API 服务的域名。

版权所有：腾讯云计算（北京）有限责任公司

第185 共732页

API服务管理 默认项目

创建API服务

请输入API服务名称

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test1aaa 1.2.1	开发中		http://100000076228-914.api.ipa...	2022-01-05 14:25:08		0		0	编辑 添加endpoint 更多操作
API_test2 1.0.1	开发中	0	http://100000076228-137.api.ipa...	2021-12-30 15:22:41		1		0	编辑 添加endpoint 更多操作
API_testYT 1.0.1	上线	0	https://100000076228-155.api.ipa...	2021-12-30 15:22:41		1	BasicAuth/OAuth2.0	2	查看 添加endpoint 更多操作
API_test11 1.0.0	下线	AA的标签	http://100000076228-770.api.ipa... https://100000076228-770.api.ipa...	2021-12-30 15:22:41		1	BasicAuth	0	查看 添加endpoint 更多操作
API_test1 1.0.5	开发中		http://100000076228-797.api.ipa...	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作

共 5 条

10 条 / 页

2. 在 API Endpoint 列表中获取需要请求的 Endpoint 路径。

API服务管理 默认项目

创建API服务

请输入API服务名称

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test1aaa 1.2.1	开发中		http://100000076228-914.api.ipa...	2022-01-05 14:25:08		0		0	编辑 添加endpoint 更多操作
API_test2 1.0.1	开发中	0	http://100000076228-137.api.ipa...	2021-12-30 15:22:41		1		0	编辑 添加endpoint 更多操作
API_testYT 1.0.1	上线	0	https://100000076228-155.api.ipa...	2021-12-30 15:22:41		1	BasicAuth/OAuth2.0	2	查看 添加endpoint 更多操作

访问路径	调用方法	分组	路径地址	后端服务	最近修改时间	操作
/test	POST	0	https://100000076228-155.api.ipa...	集成流/HelloWorldtest/HelloWorld_Flow/天津	2021-11-18 11:3...	查看 删除 调试

共 1 条

10 条 / 页

API服务名称	状态	分组	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test11 1.0.0	下线	AA的标签	http://100000076228-770.api.ipa... https://100000076228-770.api.ipa...	2021-12-30 15:22:41		1	BasicAuth	0	查看 添加endpoint 更多操作
API_test1 1.0.5	开发中		http://100000076228-797.api.ipa...	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作

共 5 条

10 条 / 页

3. 将 Endpoint 路径拼接在 API 服务域名后面，即可获取完整的 API 请求域名。请将此域名保存，步骤4 中会用到。

步骤3：获取用户请求 API 的权限

1. 在 API 用户中心 首页单击用户管理。

API用户中心 默认项目

审批管理 用户管理

请输入关键词搜索

服务名称	版本	鉴权方式	用户数	最近更新时间	操作
API_test1	v1.0.1	BasicAuth/OAuth2.0	0	2021-11-15 15:18:00	查看

共 1 条

10 条 / 页

2. 单击添加用户，并配置以下信息。

添加用户

基本信息

用户名称 *

服务名称 *

用户密码 *

联系邮箱 *

联系电话 *

所属企业

策略信息

开启访问限制

请求频率策略

-

100

+

次/

日

请输入请求频率

访问限制策略

-

100

+

次/

日

请输入访问限制

确认

关闭

3. 帮助用户填写必要信息，并选择需要访问的 API 服务名称。

用户管理

添加

成员名称	状态 ▾	账号ID	用户邮箱	用户电话
asd	禁用	3		
dsa	审批通过	7		
ddd	审批通过	8		
test1	审批通过	73		
yetong	审批通过	79		
	审批通过	108		
新用户test	审批通过	334		

共 7 条

添加用户

基本信息

用户名称 *

test11

服务名称 *

API_test2

API_test1aaa

API_test2

API_testYT

API_test11

API_test1

用户密码 *

联系邮箱 *

联系电话 *

所属企业

策略信息

开启访问限制

确认

关闭

4. 绑定好 API 服务之后，返回用户管理列表获取此用户的 OAuth 信息（当前系统会默认给添加的用户进行自动审批，所以无需手动审批操作），并复制保存 OAuth 的 Token 获取地址、Client ID、Client Secret 等信息，连同用户的账号和密码一同分享给用户。

←

用户管理-API_test1

添加

请输入关键词搜索

成员名称	状态 ▾	账号ID	用户邮箱	用户电话	操作
test1	审批通过	73			OAuth信息 禁用 启用 查看详细信息

共 1 条

10

条 / 页

1

/ 1 页

步骤4: 从用户侧调用 API (以 postman 为例)

- API 服务无需验证的情况:

POST

oud.com/

Params

Send

Save

Authorization

Headers (2)

Body

Pre-request Script

Tests

Code

Type

No Auth

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 117 ms

Pretty

Raw

Preview

Text

1

- API 服务需要 Basic Auth 的情况:

POST

:om/

Params

Send

Save

Authorization

Headers (2)

Body

Pre-request Script

Tests

Code

Type

Basic Auth

Clear

Update Request

Username

t

Password

☒ Show Password

The authorization header will be generated and added as a custom header

☐ Save helper data to request

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 124 ms

- API 服务需要 OAuth2.0 的情况:

1.1 在用户信息处获取 Client ID 和 Client Secret，在待访问的 API 服务页面获取 Access Token URL。

获取 OAuth 信息

← 用户管理-API_test1

添加

成员名称	状态 ▾	账号ID	用户邮箱	用户电话
test1 10	审批通过	73 10	test@test.com	18888888888

共 1 条

用户详情

基本信息

用户名

test1

服务名称

▼

用户密码

..... 10

联系邮箱

test@test.com

联系电话

18888888888

所属企业

test

验证信息

Client ID

30 10

Client Secret

10 10

策略信息

开启访问限制

☐

获取 token 地址

API服务管理

默认项目

帮助文档

创建API服务

请输入API服务名称

API服务名称	状态	分组 ▾	服务域名	更新时间	数据统计	endpoint统计	鉴权方式	用户数量	操作
API_test1aaa 1.2.1	开发中		http://100000076228-914.apl.pa... 10	2022-01-05 14:25:08		0		0	编辑 添加endpoint 更多操作 ▾
API_test2 1.0.1	开发中	0	http://100000076228-137.apl.pa... 10	2021-12-30 15:22:41		1		0	编辑 添加endpoint 更多操作 ▾
API_testYT 1.0.1	上线	0	https://100000076228-155.apl.pa... 10	2021-12-30 15:22:41		1	BasicAuth/OAuth2.0	2	查看 endpoint 更多操作 ▾
API_test11 1.0.0	下线	AA的标签	http://100000076228-770.apl.pa... 10 https://100000076228-770.apl.pa... 10	2021-12-30 15:22:41		1	BasicAuth	0	查看 endpoint 更多操作 ▾
API_test1 1.0.5	开发中		http://100000076228-797.apl.pa... 10	2021-12-30 15:22:41		0		1	编辑 添加endpoint 更多操作 ▾

共 5 条

10 条 / 页

1 / 1 页

token 地址如下：

验证信息

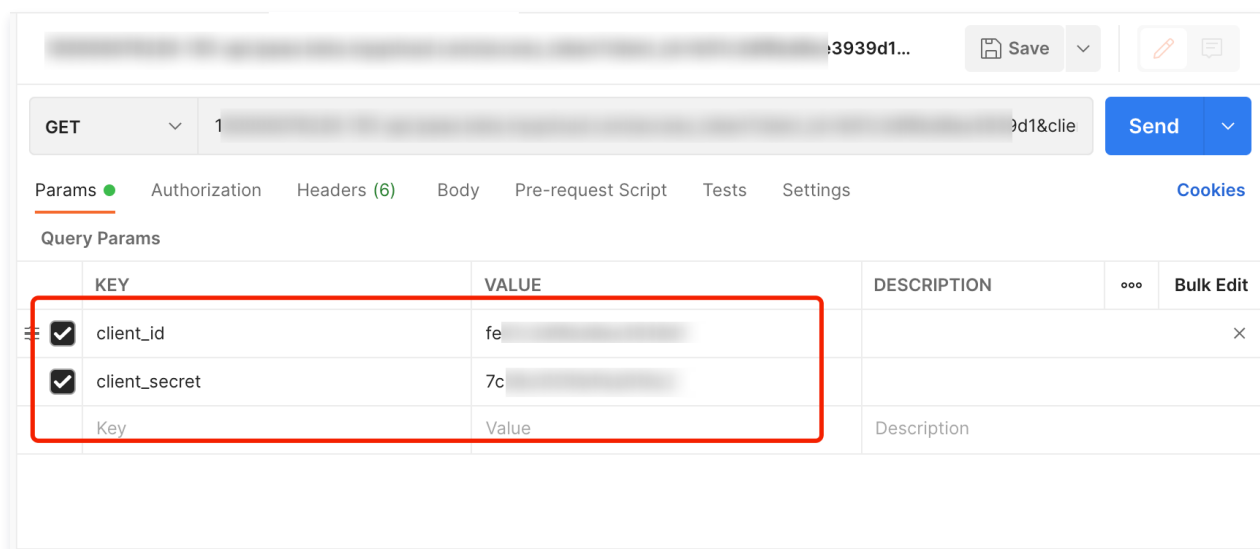
Token获取地址 https://100000076228-155.apl.pa...m/access_token

Refresh Token获取地址 https://100000076228-155.apl.pa...ash_token

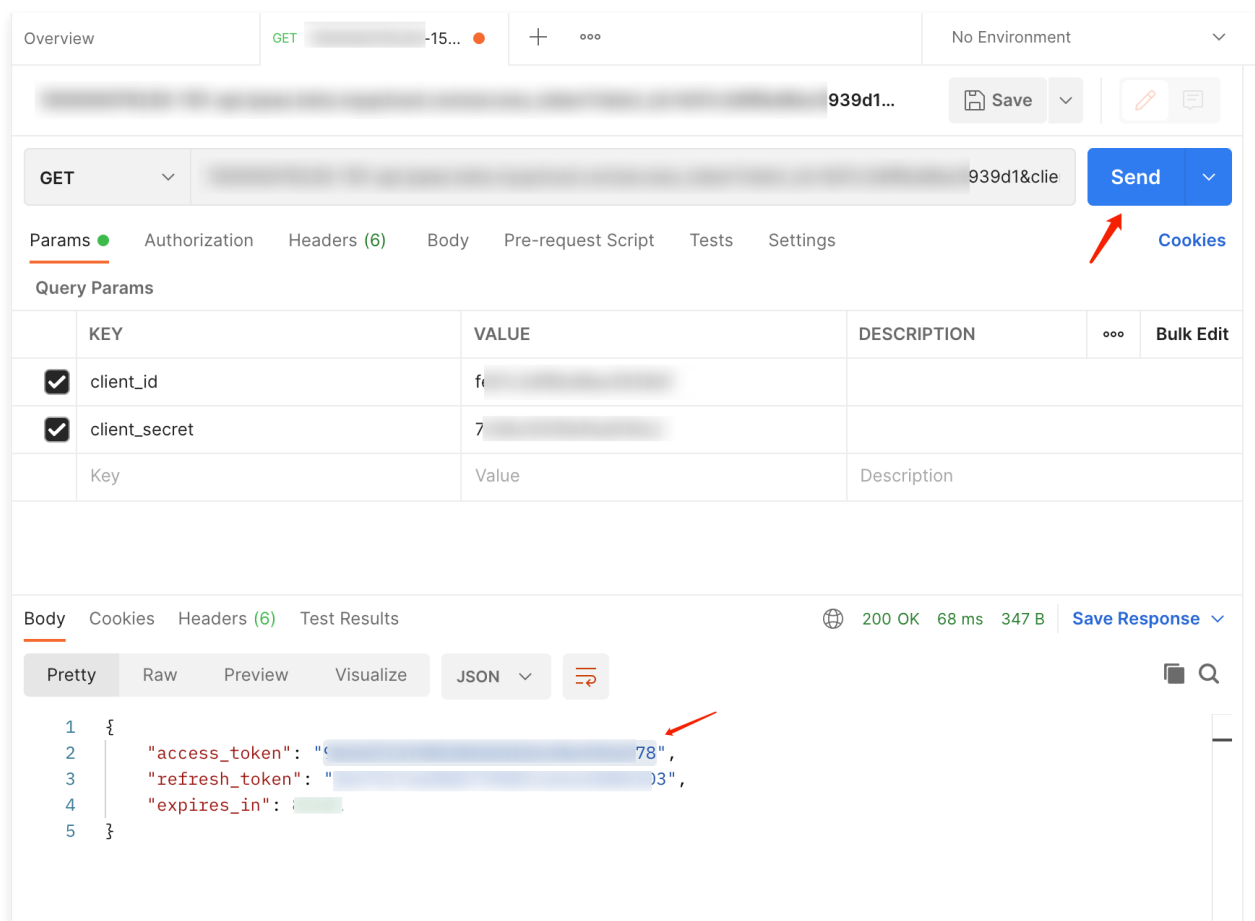
Token过期时间 24h

1.2 复制之后，在 postman 中创建一个新的请求，填写入上方的 token 获取地址并使用“GET”方法（此处的鉴权方式应为“NoAuth”）。随后选择 Params 标签页，输入 client 信息。输入方式：

- 第一列：key 为 client_id，value 为从上方复制的 Client ID 对应的内容。
- 第二列：key 为 client_sectet，value 为从上方复制的 Client Secret 对应的内容。



1.3 单击 **send**，从界面下方的“body”处复制“access_token”字段的值。此即为本次调用时需要用到的 Token 信息。



1.4 重新打开一个请求界面，填入需要请求的 API 域名，并选择 Bearer Token 模式。在右侧的“Token”处输入前面获得的 Token，单击 **send**，即可看到访问结果。

The screenshot displays the Tencent Cloud API Explorer interface. At the top, the request method is set to **POST** and the URL is `https://[redacted].com/test`. The **Authorization** tab is selected, showing a **Bearer ...** token type. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. [Learn more about variables](#)". The **Token** field contains a partially visible value starting with `9da`. Below the request configuration, the **Body** tab is active, showing a JSON response in **Pretty** format: `{ "Hello": "World" }`. The status bar at the bottom indicates a **200 OK** response with a response time of **417 ms** and a size of **296 B**. Red arrows highlight the URL, the token field, and the JSON response.

运维中心

监控管理

最近更新时间：2023-06-08 14:47:58

以下视频将为您介绍运行监控模块的各个功能查看方式：

[观看视频](#)

操作场景

运行监控集合了集成应用维度、集成流、连接器维度、组件维度的基本运行情况概览及错误管理的数据上报，您可以查看某个集成流在一段时间内的实时及历史运行情况。

关于监控指标中各参数的说明请参见 [监控指标名词解释](#)。

说明

目前腾讯云数据连接器的监控管理以项目维度划分，查看日志请先选择对应项目。

监控管理

测试

错误管理

定时触发

20220831_2.0

全部流

天津

近1天

搜索

操作步骤

监控管理

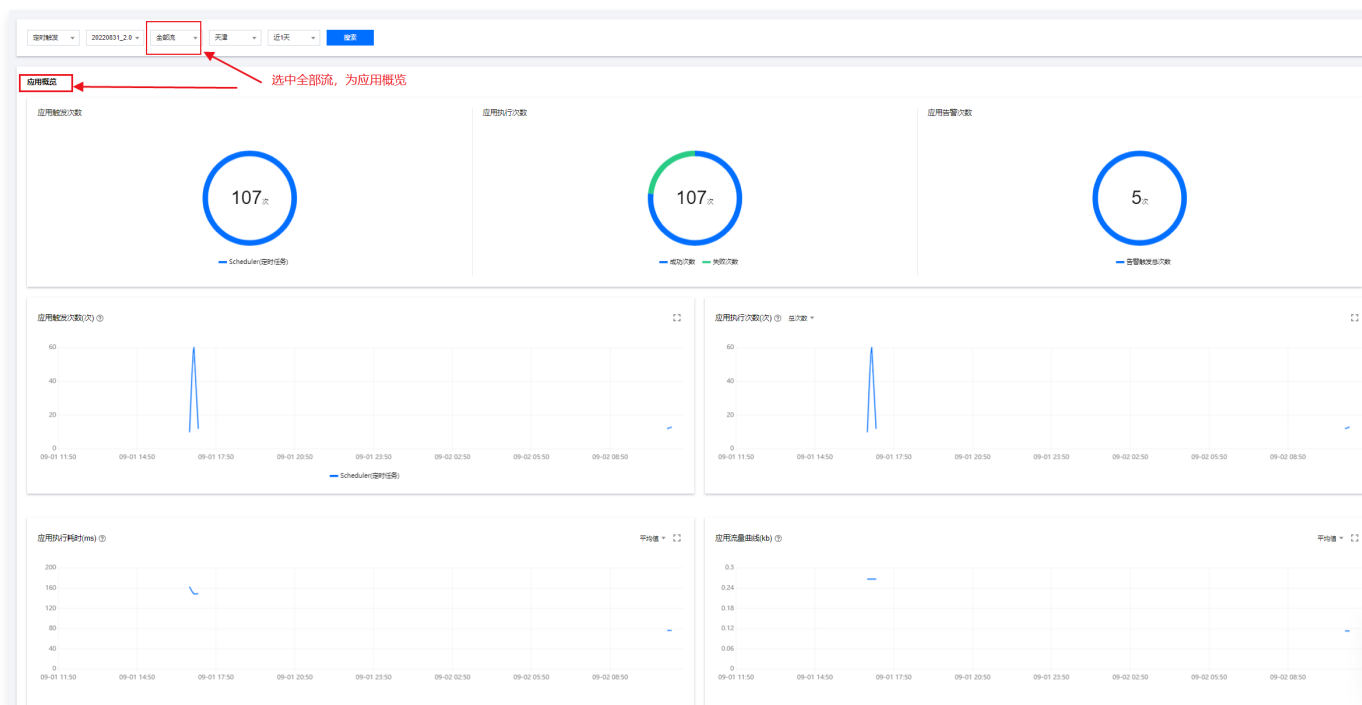
支持查看实时和历史监控数据，可以通过监控数据分析对应流的运行情况。默认展示当天集成应用的监控概览，若需要查看历史数据，可自定义时间维度后选择对应应用或流点击搜索进行查看。默认展示最近应用的近一天的概览情况。

注意

目前控制台支持检索近30天内的监控。若有检索更早期日志的需求，请 [提交工单](#) 联系我们。

应用概览数据查询

- 登录 [腾讯云数据连接器控制台](#) > [监控管理](#) > [监控管理](#)。
- 在[监控管理](#)页面，默认选中全部流即为应用概览数据，支持查看某个应用的应用触发次数、应用执行次数、应该告警次数、应用执行耗时（ms）、应用流量曲线（kb）等应用维度指标。



搜索应用监控数据说明：支持按应用名称、版本、集成流（必选全部流）、运行环境、时间等维度来进行搜索。

单击时间下拉，将显示设置搜索时间的面板，支持近5分钟、近15分钟、近1小时、近6小时、近1天、近3天、近7天、今天、昨天、前天、本周、自定义时间区间的时间设置。通过预定义的常用的时间段可快速此区间的日志。

监控管理

监控管理

错误管理

cdc测试

20220815_1.0

全部流

天津

近1天

搜索

应用概览

应用触发次数

最近时间

相对时间

自定义时间

2022-08-17 10:03:10 - 2022-08-18 10:03:10

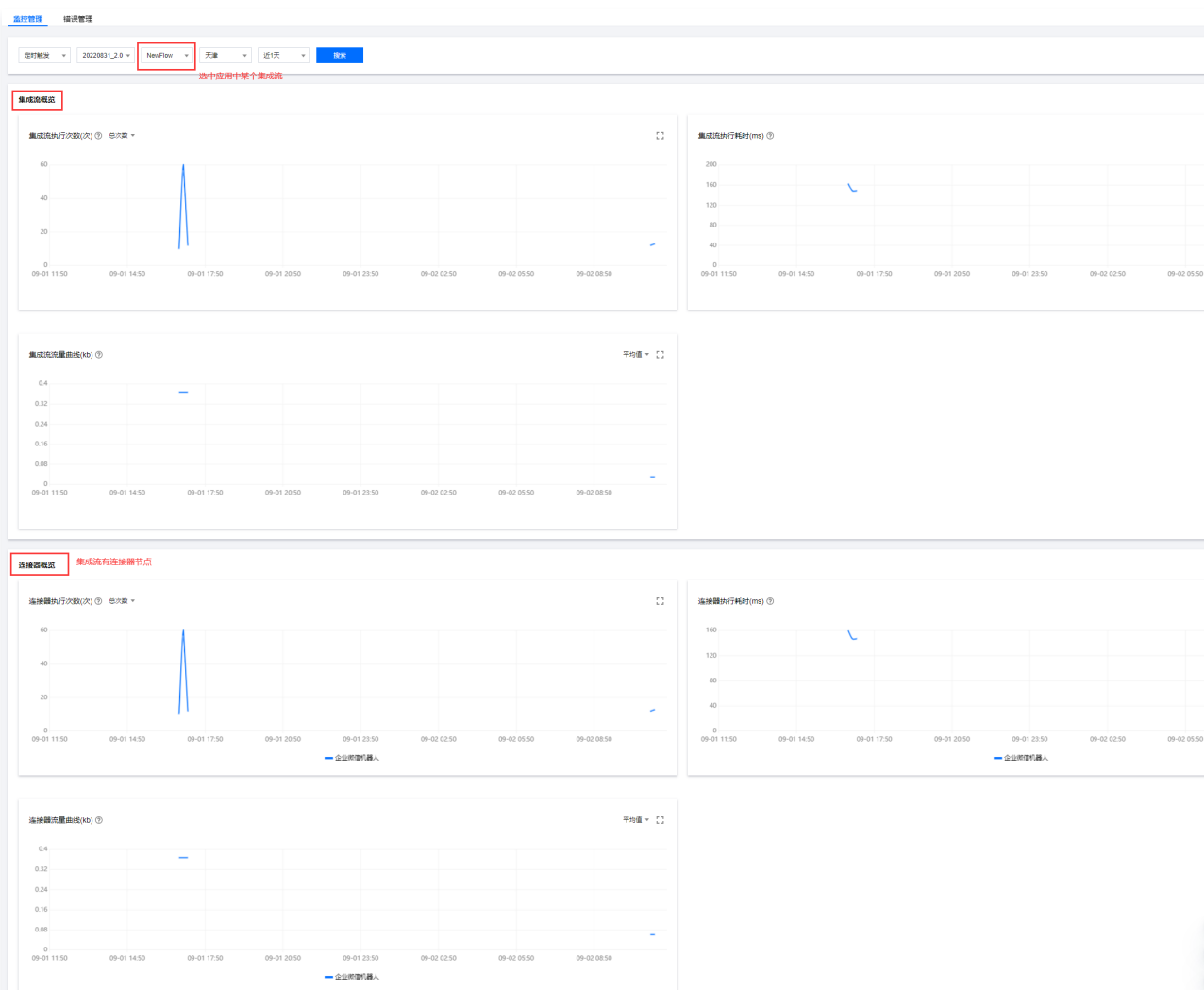
仅支持查看近7天的监控数据

确定 取消

集成流数据查询

- 登录 [腾讯云数据连接器控制台](#) > [监控管理](#) > [监控管理](#)。
- 在[监控管理](#)页面，选择应用中的某条集成流即可查询集成流概览数据及连接器概览数据。
 - 集成流概览：支持查看某个集成流的集成流执行次数（总次数/成功次数/失败次数）、集成流执行耗时（ms）、集成流流量曲线（kb）等集成流维度指标。

- 连接器概览：支持查看集成流中连接器的连接器执行次数（总次数/成功次数/失败次数）、连接器执行耗时（ms）、连接器流量曲线（kb）等连接器维度指标。



搜索集成流监控数据说明：支持按项目、应用、版本、集成流（必选某个集成流）、运行地域、时间等维度来搜索查看指定集成流的集成流概览和连接器概览。

单击时间下拉，将显示设置搜索时间的面板，支持近5分钟、近15分钟、近1小时、近6小时、近1天、近3天、近7天、今天、昨天、前天、本周、自定义时间

区间的时间设置。通过预定义的常用的时间段可快速此区间的日志。

监控管理

监控管理

错误管理

cdc测试

20220815_1.0

全部流

天津

近1天

搜索

应用概览

应用触发次数

最近时间

近5分钟

近15分钟

近30分钟

近1小时

近6小时

近1天

近3天

近7天

相对时间

今天

昨天

前天

本周

自定义时间

2022-08-17

10 : 03 : 10

-

2022-08-18

10 : 03 : 10

仅支持查看近7天的监控数据

确定

取消

错误管理

注意

目前控制台支持检索近一个月的错误管理内容。若有检索更早期日志的需求，请 [提交工单](#) 联系我们。

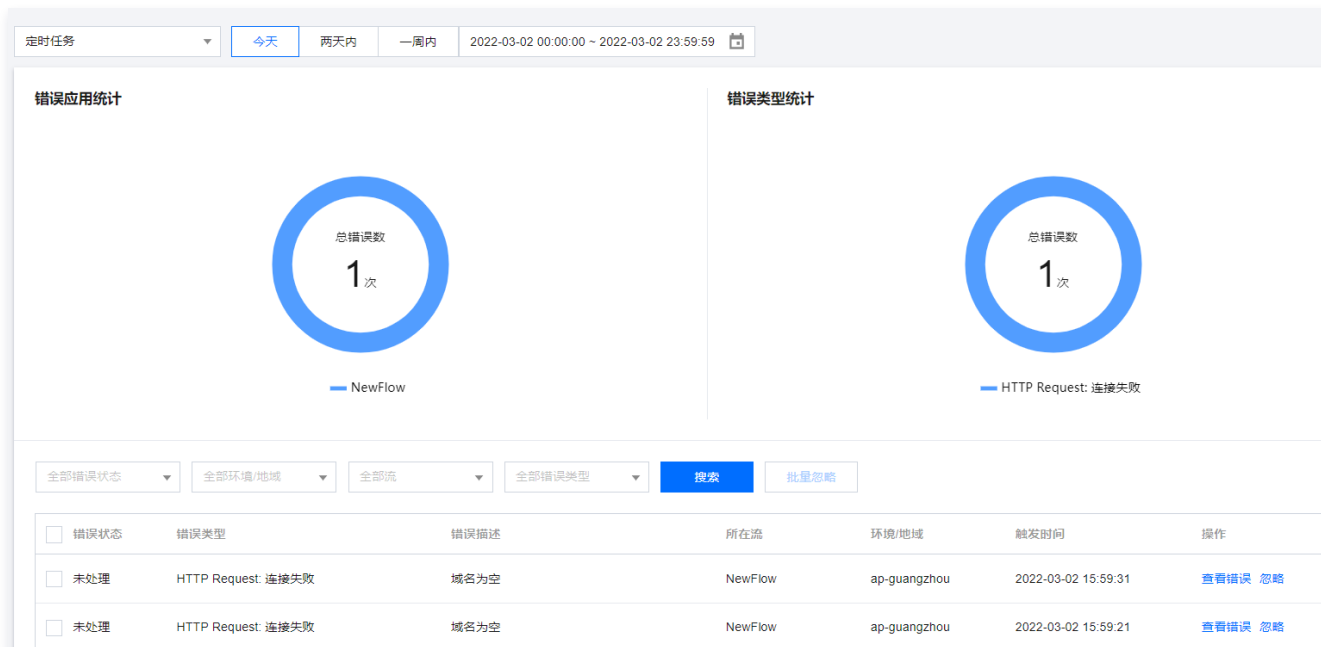
查看错误管理列表

支持对运行失败的应用的错误次数和错误类型等数据进行集中展示及快速处理。

支持通过项目、应用、错误状态、环境/地域、集成流、错误类型、时间等维度来进行搜索。列表如下：

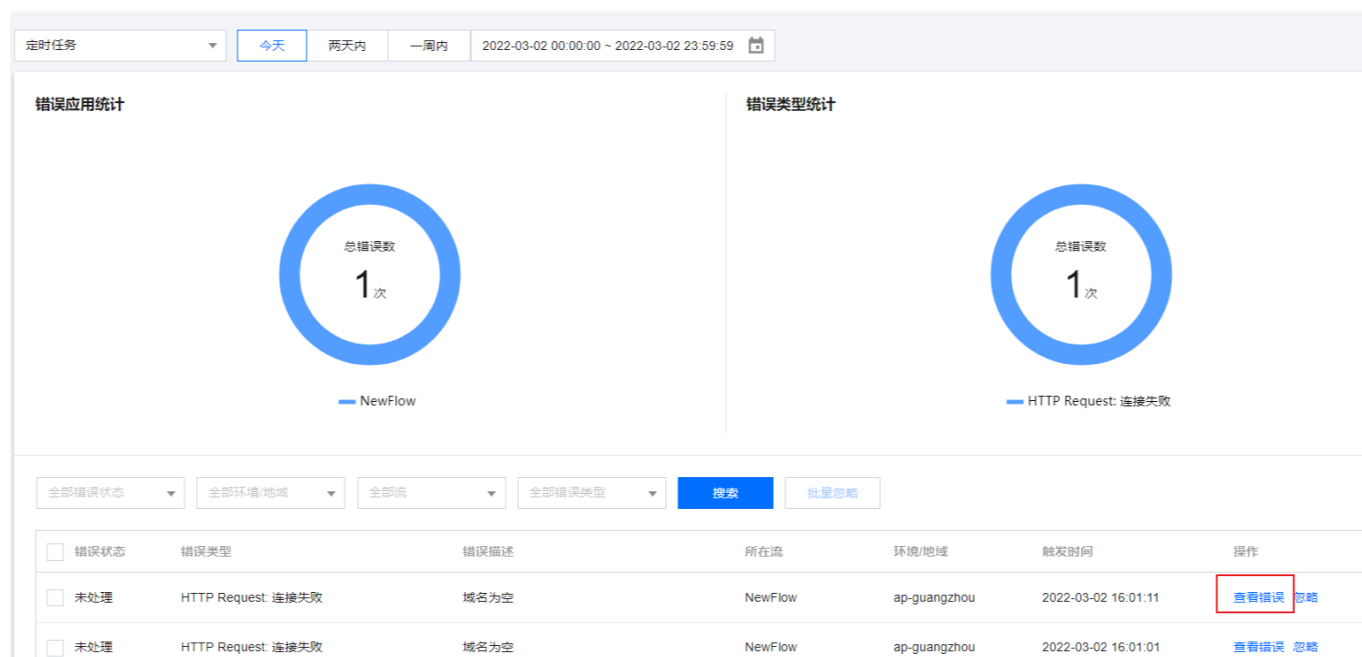
- 错误应用统计：应用运行失败，某条集成流的出现错误的总次数。
- 错误类型统计：应用运行失败，某种错误类型的出现总次数。
- 错误状态：已处理的错误不在显示，只展示未处理或已忽略。
- 错误类型：将常见的报错分为大类，方便根据错误类型筛选错误流。
- 错误描述：进一步解释报错的详细原因，方便尽快找到错误根源进行处理。
- 所在流：报错出现的集成流名称。
- 环境/地域：报错的应用所属的具体环境/地域。方便尽快定位出现错误的环境。

- 触发时间：集成流触发报错时的具体时间。



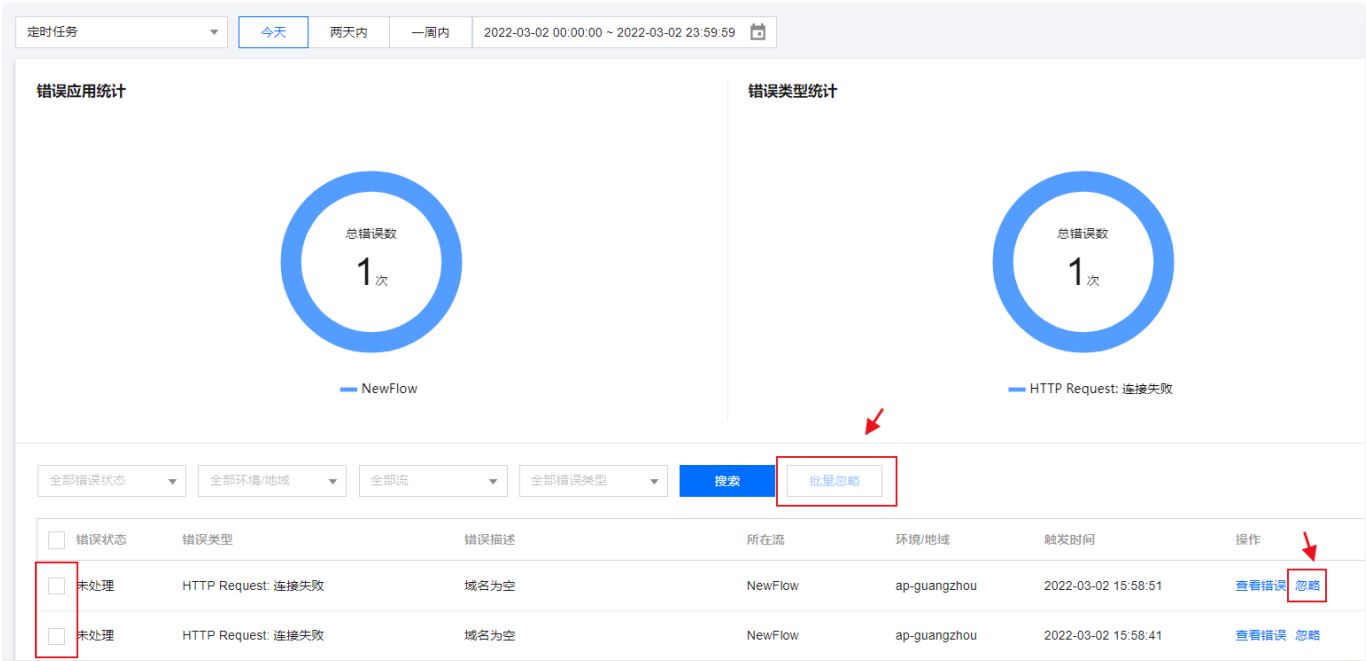
查看错误

单击查看错误将快速跳转到对应报错的集成流界面可快速定位到出现错误的集成流。



忽略/批量忽略

忽略错误的操作，若某个集成流的错误已处理完成，其余相邻时间段内的相同报错可点击对应错误条目的忽略，即可忽略此错误。或者选中对应错误条目后点击批量忽略。



监控指标名词解释

指标类别	指标名称	指标详解
应用	应用监控概览	以应用维度对数据进行聚合（选中全部流默认展示应用维度），可以查看选定应用的符合筛选条件的对应数据和图表信息。
	应用触发次数（次）	统计某应用下全部流被触发的次数之和，涵盖 Scheduler、HTTP Listener、Kafka Consumer 等触发器。触发一次会执行所引用的所有流，因此应用触发次数不超过应用执行次数。
	应用执行次数（次）	统计某应用下全部流被执行的次数之和，分为总次数、成功次数、失败次数。以“近1天（粒度10分钟）的总次数”为例，折线上每个值是指近10分钟内该应用被执行的总次数。
	应用执行耗时（ms）	统计某应用被执行所耗时间。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该应用全部耗时数据正序排列后第99.9分位的值，即99.9%的耗时数据都不超过该值。其余分位值同理。
	应用消耗流量（kb）	统计某应用被执行所耗流量。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该应用全部流量数据正序排列后第99.9分位的值，即99.9%的流量数据都不超过该值。其余分位值同理。
集成流	集成流监控概览	以流维度对数据进行聚合，可以查看选定应用的选定流的符合筛选条件的对应数据和图表信息。
	集成流执行次数（次）	统计某条流被执行的次数，分为总次数、成功次数、失败次数。以“近1天（粒度10分钟）的总次数”为例，折线上每个值是指近10分钟内该流被执行的总次数。
	集成流执行耗时（ms）	统计某条流被执行所耗时间。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该流全部耗时数据正序排列后第99.9分位的值，即99.9%的耗时数据都不超过该值。其余分位值同理。
	集成流消耗流量（kb）	统计某条流被执行所耗流量。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该流全部流量数据正序排列后第99.9分位的值，即99.9%的流量数据都不超过该值。其余分位值同理。
触发器	连接器监控概览	以连接器维度对数据进行聚合，只会统计被执行的连接器的情况，可以查看选定应用的选定流的选定连接器的符合筛选条件的对应数据和图表信息。
	连接器执行次数（次）	统计某条流下各连接器被执行的次数，分为总次数、成功次数、失败次数。以“近1天（粒度10分钟）的总次数”为例，折线上每个值是指近10分钟内该连接器被执行的总次数。
	连接器执行耗时（ms）	统计某条流下各连接器被执行所耗时间。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该连接器全部耗时数据正序排列后第99.9分位的值，即99.9%的耗时数据都不超过该值。其余分位值同理。
	连接器消耗流量（kb）	统计某条流下各连接器被执行所耗流量。以“近1天（粒度10分钟）的P99.9分位值”为例，折线上每个值是将近10分钟内该连接器全部流量数据正序排列后第99.9分位的值，即99.9%的流量数据都不超过该值。其余分位值同理。

运行日志

最近更新时间：2023-06-08 14:47:58

以下视频将为您介绍运行监控模块的各个功能查看方式：

[观看视频](#)

操作场景

运行日志集合了集成应用基本运行日志详情、定时任务管理（以 scheduler 为触发器的流），您可以查看某个集成流在一段时间内的实时及历史运行日志情况。

说明

目前腾讯云数据连接器的运行日志以项目维度划分，查看日志前请先选择对应项目。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近30分钟

搜索

日志数量 100条

运行时间	日志级别	作业标识
2022-09-02 11:12:00.871	● ERROR	f04833d3-f546-0137-af36-71e9db25c69e

操作步骤

运行日志

步骤1: 查询运行日志

运行日志支持查看历史日志和实时日志，可以通过日志信息回溯一次作业的运行情况和消息数据。

注意

目前控制台支持检索近一个月的运行日志。若有检索更早期日志的需求，请 [提交工单](#) 联系我们。

- 登录腾讯云数据连接器控制台，进入运行日志 > 运行日志 页面。
- 在 运行日志 页面，默认展示最近应用的近30分钟的运行日志，日志按就近原则进行排序，日志列表包括：
 - 运行时间：产生日志的时间。
 - 日志级别：不同级别的日志采用不同颜色的图标记。
 - 作业标识：产生日志时运行的标记，通过作业标识可将一次运行的日志全部找出来，便于分析。
 - 集成流名称：产生日志的集成流。
 - 组件/连接器：产生日志的组件或连接器。
 - 操作名称：产品日志时的操作。
 - 日志类别：如果是 Logger 组件输出的日志，记录用户自定义的日志类别，其它组件或连接器则记录默认的输出事件。
 - 环境：应用运行的环境。
 - 日志内容：日志内容摘要。
 - 搜索日志说明：支持按应用名称、版本、集成流名称、运行环境、日志级别、作业标识、日志内容、时间等维度来进行搜索。

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近1天

搜索

日志数量 1126条

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-01 17:11:50.862	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:50.880	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:50.877	INFO	1c8848a3-caf3-0564-87b8-304376818320	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:50.548	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:40.279	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:40.277	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:40.274	INFO	4776db68-6964-fb3-69db-efcfdaa2246	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:40.137	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:30.591	INFO	7796b896-140d-40ed-e209-52c9fe7ed35	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:30.589	INFO	7796b896-140d-40ed-e209-52c9fe7ed35	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:30.585	INFO	7796b896-140d-40ed-e209-52c9fe7ed35	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:30.450	INFO	7796b896-140d-40ed-e209-52c9fe7ed35	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:20.807	INFO	b857356-334e-e8d5-7564-a2c3f7ebdb08	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:20.805	INFO	b857356-334e-e8d5-7564-a2c3f7ebdb08	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:20.801	INFO	b857356-334e-e8d5-7564-a2c3f7ebdb08	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:20.879	INFO	b857356-334e-e8d5-7564-a2c3f7ebdb08	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:10.520	INFO	934b4e3-9d77-6493-e288-f9ec33bc021c	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:10.519	INFO	934b4e3-9d77-6493-e288-f9ec33bc021c	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:10.516	INFO	934b4e3-9d77-6493-e288-f9ec33bc021c	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:10.359	INFO	934b4e3-9d77-6493-e288-f9ec33bc021c	NewFlow	timer	-	app start	天津	日志内容

3. 单击时间下拉，将显示设置搜索时间的面板，支持近5分钟、近15分钟、近1小时、近6小时、近1天、近3天、近7天、近30天、今天、昨天、前天、本周、上周、自定义时间区间的时间设置。通过预定义的常用的时间段可快速此区间的日志。
- 支持自动刷新日志的能力：单击自动刷新开关将显示刷新时间间隔时间。设置自动刷新时间后将根据时间间隔时间不断刷新日志。

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近1天

搜索

日志数量 1126条

是否自动刷新

刷新间隔(10秒)

刷新间隔(15秒)

刷新间隔(30秒)

确定

取消

② 设置刷新时间间隔时间

① 启动自动刷新

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-01 17:11:50.862	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:50.880	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:50.877	INFO	1c8848a3-caf3-0564-87b8-304376818320	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:50.548	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:40.279	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:40.277	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:40.274	INFO	4776db68-6964-fb3-69db-efcfdaa2246	do-request	http request	-	app outbound	天津	日志内容

- 查看日志详情：单击某日志内容，右侧展示日志的详情。

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近1天

搜索

日志数量 1126条

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-01 17:11:50.862	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:50.880	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:50.877	INFO	1c8848a3-caf3-0564-87b8-304376818320	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:50.548	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:40.279	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:40.277	INFO	4776db68-6964-fb3-69db-efcfdaa2246	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:40.274	INFO	4776db68-6964-fb3-69db-efcfdaa2246	do-request	http request	-	app outbound	天津	日志内容

日志详情

运行时间

2022-09-01 17:11:50.862

日志级别

INFO

作业标识

1c8848a3-caf3-0564-87b8-304376818320

操作名称

定时触发

集成流名称

NewFlow

组件/连接器

timer

操作名称

-

日志类别

app end

日志内容

com.tencent.tier.subject.time:msg@flow-376, res:1134, remaining:0

步骤2：通过作业标识快速搜索

版权所有：腾讯云计算（北京）有限责任公司

第200 共732页

集成流运行一次产生一个作业标识，通过作业标识来标记集成流在一次运行过程中产生的日志，通过作业标识来快速搜索出集成流一次运行中所有生产的日志。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近1天

搜索

日志数量 11条

上一步

下载日志

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-01 17:11:50.682	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:50.680	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:50.677	INFO	1c8848a3-caf3-0564-87b8-304376818320	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:50.548	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app start	天津	日志内容
2022-09-01 17:11:40.279	INFO	4776db68-6964-f8b3-e4c0-efc0a8a2f46	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:40.277	INFO	4776db68-6964-f8b3-e4c0-efc0a8a2f46	NewFlow	企业微信机器人	send	app outbound	天津	日志内容

快速搜索出一次运行的所有日志。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部日志级别

1c8848a3-caf3-0564-87b8-30437

请输入日志内容关键词

近1天

搜索

日志数量 4条

上一步

下载日志

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-01 17:11:50.682	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app end	天津	日志内容
2022-09-01 17:11:50.680	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-01 17:11:50.677	INFO	1c8848a3-caf3-0564-87b8-304376818320	do-request	http request	-	app outbound	天津	日志内容
2022-09-01 17:11:50.548	INFO	1c8848a3-caf3-0564-87b8-304376818320	NewFlow	timer	-	app start	天津	日志内容

已加载更多所有内容

步骤3：重试集成任务

在发布上线后，为了给集成项目保驾护航，腾讯云数据连接器已完善集成流监控指标，增加可靠性分析能力。当触发集成任务发布失败时允许自动重试或者用户手动重试，避免数据丢失。

- 自动重试：后台自动执行，若自动重试成功则更新控制台此条日志状态为成功，不记录之前失败的日志。
- 手动重试：若自动重试的最终结果是失败，则会更新控制台此条日志状态为失败，允许用户手动重试。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近30分钟

搜索

日志数量 28条

上一步

下载日志

运行时间	日志级别	作业标识	集成流名称	组件/连接器	操作名称	日志类别	环境	操作
2022-09-02 11:09:10.643	ERROR	50a02955-a226-52ed-006c-3c02c8c9059e	NewFlow	timer	-	app end	天津	日志内容 重试 下载
2022-09-02 11:09:10.519	ERROR	50a02955-a226-52ed-006c-3c02c8c9059e	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-02 11:09:10.518	ERROR	50a02955-a226-52ed-006c-3c02c8c9059e	do-request	http request	-	app outbound	天津	日志内容
2022-09-02 11:09:10.516	INFO	50a02955-a226-52ed-006c-3c02c8c9059e	NewFlow	timer	-	app start	天津	日志内容
2022-09-02 11:09:00.114	ERROR	8b95dbde-9487-f10a-c19e-59a1b6114488	NewFlow	timer	-	app end	天津	日志内容 重试 下载
2022-09-02 11:09:00.054	ERROR	8b95dbde-9487-f10a-c19e-59a1b6114488	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-02 11:09:00.054	ERROR	8b95dbde-9487-f10a-c19e-59a1b6114488	do-request	http request	-	app outbound	天津	日志内容
2022-09-02 11:09:00.052	INFO	8b95dbde-9487-f10a-c19e-59a1b6114488	NewFlow	timer	-	app start	天津	日志内容
2022-09-02 11:08:59.291	ERROR	64cc087f-f3b6-368e-e4d770b6d82cf	NewFlow	timer	-	app end	天津	日志内容 重试 下载
2022-09-02 11:08:59.232	ERROR	64cc087f-f3b6-368e-e4d770b6d82cf	NewFlow	企业微信机器人	send	app outbound	天津	日志内容
2022-09-02 11:08:59.231	ERROR	64cc087f-f3b6-368e-e4d770b6d82cf	do-request	http request	-	app outbound	天津	日志内容
2022-09-02 11:08:59.229	INFO	64cc087f-f3b6-368e-e4d770b6d82cf	NewFlow	timer	-	app start	天津	日志内容
2022-09-02 11:08:40.836	ERROR	08ba0975-2c96-082f-4acc-80750a61176	NewFlow	timer	-	app end	天津	日志内容 重试 下载

步骤4：下载日志

运行日志提供日志下载功能。单击[下载日志](#)图标即可下载日志。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部日志级别

请输入作业标识

请输入日志内容关键词

近30分钟

搜索

日志数量 28条

运行时间

日志级别

作业标识

集成流名称

组件/连接图

操作名称

日志类别

环境

操作

2022-09-02 11:09:10.643	ERROR	50a02955-a228-52ed-08fc-3c02dc909de	NewFlow	timer	-	app-end	天津	日志内容 重试 下载	
2022-09-02 11:09:10.519	ERROR	50a02955-a228-52ed-08fc-3c02dc909de	NewFlow	企业微信机器人	send	app-outbound	天津	日志内容	
2022-09-02 11:09:10.518	ERROR	50a02955-a228-52ed-08fc-3c02dc909de	NewFlow	do-request	http-request	-	app-outbound	天津	日志内容
2022-09-02 11:09:10.516	INFO	50a02955-a228-52ed-08fc-3c02dc909de	NewFlow	timer	-	app-start	天津	日志内容	
2022-09-02 11:09:00.114	ERROR	8bf6dbde-9487-f10a-c19e-59a1b0b14488	NewFlow	timer	-	app-end	天津	日志内容 重试 下载	
2022-09-02 11:09:00.054	ERROR	8bf6dbde-9487-f10a-c19e-59a1b0b14488	NewFlow	企业微信机器人	send	app-outbound	天津	日志内容	
2022-09-02 11:09:00.054	ERROR	8bf6dbde-9487-f10a-c19e-59a1b0b14488	NewFlow	do-request	http-request	-	app-outbound	天津	日志内容
2022-09-02 11:09:00.052	INFO	8bf6dbde-9487-f10a-c19e-59a1b0b14488	NewFlow	timer	-	app-start	天津	日志内容	

上一步

下载日志

定时任务管理

对于使用了 Scheduler 组件，根据制定规则定时触发集成流的应用，支持查看30天内流运行详情，包括集成应用的触发时间、结束时间、状态及对应操作的详细列表。

注意

目前控制台支持检索30天内定时任务内容。若有检索更早期日志的需求，请 [提交工单](#) 联系我们。

1. 登录腾讯云数据连接器控制台，进入[运行日志](#) > [定时任务](#) 页面。
2. 在 [定时任务](#) 页面，默认展示最近应用的近30分钟的定时任务运行日志，日志按就近原则进行排序，日志列表包括：
- 触发时间：触发应用的时间。

○ 结束时间：触发完毕的时间。

○ 状态：执行成功/失败的状态。

○ 作业标识：产生日志时运行的标记，通过作业标识可将一次运行的日志全部找出来，便于分析。

○ 集成流名称：产生日志的集成流。

○ 环境：应用运行的环境。

○ 操作：查看日志详情。

运行日志

定时任务

定时触发

全部版本

全部流

天津

全部任务状态

请输入作业标识

近30分钟

搜索

触发时间	结束时间	状态	作业标识	集成流名称	环境	操作
2022-09-02 11:12:00.726	2022-09-02 11:12:00.871	失败	04833a5c-f548-9137-af06-71e9db2c5d9e	NewFlow	天津	查看详情
2022-09-02 11:11:50.312	2022-09-02 11:11:50.396	失败	33a05fba-4391-6434-c596-a050b9e07806	NewFlow	天津	查看详情
2022-09-02 11:11:40.201	2022-09-02 11:11:40.291	失败	47277046-0196-0905-6ac1-4737002d9ec2	NewFlow	天津	查看详情
2022-09-02 11:11:30.06	2022-09-02 11:11:30.151	失败	674ae6a0-1d54-9063-2173-8599502095d	NewFlow	天津	查看详情
2022-09-02 11:11:20.541	2022-09-02 11:11:20.597	失败	4e038f130-8061-6430-e03c-691f5648e6a	NewFlow	天津	查看详情
2022-09-02 11:11:10.587	2022-09-02 11:11:10.638	失败	836c3d6d-6280-2a20-c086-565fc000fec	NewFlow	天津	查看详情
2022-09-02 11:11:00.217	2022-09-02 11:11:00.277	失败	c0a2a2b0-d99a-31d5-a05a-2be7d7809a21	NewFlow	天津	查看详情
2022-09-02 11:10:50.407	2022-09-02 11:10:50.471	失败	c7cc50df-fcd1-4db0-3d83-1703923882ad	NewFlow	天津	查看详情
2022-09-02 11:10:40.45	2022-09-02 11:10:40.503	失败	6999892b-7089-8807-651a-90f9a9e1bd6f	NewFlow	天津	查看详情
2022-09-02 11:10:30.574	2022-09-02 11:10:30.645	失败	58027774-80ec-ca11-7995-684425850671	NewFlow	天津	查看详情
2022-09-02 11:10:20.383	2022-09-02 11:10:20.444	失败	ca78c491-af32-5a32-ec0d-4e48d08be754	NewFlow	天津	查看详情
2022-09-02 11:10:10.085	2022-09-02 11:10:10.067	失败	665170ba-b159-e90b-9162-0aa0ba36006	NewFlow	天津	查看详情
2022-09-02 11:10:00.517	2022-09-02 11:10:00.572	失败	3cd05ccb-67e3-b3a0-12a0-2a7912c71cdd	NewFlow	天津	查看详情

- 搜索日志说明：支持按应用名称、版本、集成流名称、运行环境、作业标识、时间等维度来进行搜索。
3. 单击时间下拉，将显示设置搜索时间的面板，支持近5分钟、近15分钟、近1小时、近6小时、近1天、近3天、近7天、近30天、今天、昨天、前天、本周、上周、自定义时间区间的时间设置。通过预定义的常用的时间段可快速此区间的日志。

运行日志

V24版本测试

运行日志

定制任务

定时触发

全部版本

全部流

天津

全部任务状态

请输入作业标识

本流

搜索

最近时间

近5分钟

近15分钟

近30分钟

近1小时

近6小时

相对时间

近1天

近3天

近7天

近30天

本周

上月

自定义时间

2022-09-29 00:00 - 2022-09-04 23:59

仅支持设置最近30天的定时任务

确定

取消

作业标识	流域名名称	环境	操作
104833d3-1546-0137-af36-71e9db25c89e	NewFlow	天津	查看详情
33d05fba-4391-0434-c5f6-eb59b07806	NewFlow	天津	查看详情
47277046-0196-0965-8ac1-473700209ec2	NewFlow	天津	查看详情
b774e6a5-1d54-9da3-2173-85985c0296dd	NewFlow	天津	查看详情
ee2db130-8f01-643b-e92c-691f5648e9a	NewFlow	天津	查看详情
83dc3ded-e28b-2a2b-c885-5b5fc00f9ec	NewFlow	天津	查看详情
c09a2a2b-df9e-31d5-aa5a-2ba7d7808a21	NewFlow	天津	查看详情
c7cc50c1-fd01-4d0b-3d63-1703523862ad	NewFlow	天津	查看详情
6b99892b-7088-8887-451a-99f9e9a1ba0f	NewFlow	天津	查看详情

2022-09-02 11:10:50.407

2022-09-02 11:10:50.471

失败

c7cc50c1-fd01-4d0b-3d63-1703523862ad

2022-09-02 11:10:40.45

2022-09-02 11:10:40.583

失败

6b99892b-7088-8887-451a-99f9e9a1ba0f

告警配置

告警策略

最近更新时间：2025-05-27 14:44:12

操作场景

本文指导您在数据连接器告警模块中如何创建、修改、启停、复制和删除告警策略。
告警策略针对支持的已发布的应用设置执行失败次数等指标的阈值告警，在发生异常时及时通知您采取措施。告警策略包括名称、告警对象、告警条件、告警通知模板四个必要组成部分。

基本概念

术语	定义
告警策略	由告警名称、描述（非必填）、告警条件、告警对象和告警通知模板组成。
告警名称	告警名称用于命名告警策略。
描述	告警描述用于简单定义对应告警策略的作用，放在使用时进行区分。
告警对象	支持发布过的应用。
告警条件	是指标、判断逻辑、判断条件、统计周期和持续 N 个监控周期组成的一个有语义的条件。
通知模板	多个告警策略支持一键复用模板，适用于多种场景接收告警通知，详情请参见 新建告警通知模板 。

使用限制

功能	限制
告警对象	最多可添加3个
告警条件	最多可添加4个

操作步骤

新建告警策略

- 登录 [数据连接器控制台](#)。
- 在左侧导航栏，选择[运维中心](#) > [告警配置](#) > [告警策略](#)，进入告警策略配置页面。
- 在页面顶部，选择项目。

注意：
选择项目后，只能选择对应项目下的运行中的应用作为告警对象。

告警配置

默认项目

切换项目

帮助文档

告警历史

告警策略

通知模板

新建

全部告警对象

请输入策略名称关键词

策略名称	描述	告警对象个数	告警频率	通知模板	运行状态	操作
当前列表为空，您可以 添加 一条通知策略						

共 0 条

10 条 / 页

1 / 1 页

- 单击新建，配置告警策略，配置说明如下：

配置类型	配置项	说明
------	-----	----

基本信息	策略名称	自定义策略名称。
	描述	自定义策略描述。
配置告警规则	告警对象	选择应用名称，指该告警策略绑定用户选中的应用及应用下的所有集成流。 选择发布地域，只有选中地域下的应用达到触发条件才会触发，未勾选的地域不会触发告警。
	告警条件	告警触发条件：指标、判断逻辑、判断条件、统计周期和持续 N 个周期组成的一个有语义的条件。您可以根据图表中指标变化趋势设置告警阈值。例如，指标为应用触发次数/总次数、判断逻辑为>、判断条件为10、统计周期为1分钟、持续周期为3个周期。表示：每1分钟收集一次应用触发次数/总次数的数据，若某个应用的触发次数/总次数连续3次大于10则触发告警。 告警频率：您可以为您的每一条告警规则设置重复通知策略。即当告警产生时，您可以定义告警以特定的频率重复通知。可选：5分钟、15分钟、30分钟...12小时等重复频率。
配置告警通知	通知模板	告警通知，用户创建自定义通知模板，每个告警策略可绑定一个通知模板，详情请参见 通知模板 。

5. 配置完以上信息后单击**确定**，即成功创建告警策略。

① 说明：

指标解释，可参见 [指标名词](#) 说明。

修改告警策略

1. 进入 [数据连接器控制台](#) > [告警策略](#) 列表页。

2. 找到需要修改的告警策略，单击对应的策略名称。

3. 进入管理告警策略页，在对应的区域直接修改相关信息，单击**确定**即可保存。

←

编辑告警策略

基本信息

策略名称 *

告警测试

描述

用于测试对应应用下的告警触发测试

告警规则

告警对象 *

应用 ①

发布环境

操作

kafka-c1

广州, 天津

删除

+添加告警对象

告警条件 *

指标

判断逻辑

判断条件

操作

应用触发次数

>

1

次

删除

+添加告警条件

当以上

全部

告警条件, 按统计周期

1分钟

满足

持续三个周期

ⓘ

每2小时告警一次

告警通知

通知模板 *

test通知模板01

+新建

确定

取消

告警策略启停

您能够通过告警启停功能对告警策略进行启用/停用操作，方便您在不需要使用某项告警策略时，及时停用告警策略，避免冗余消息对您的打扰，当您需要恢复告警时，也能一键快速启动告警策略。

1. 登录 [数据连接器控制台](#)。
2. 在左侧导航栏中，单击**告警配置**，再单击**告警策略**，进入管理页面。
3. 找到您需要操作的策略，在“告警启停”一列下，单击**运行状态**开关，即可启动或停止该策略的告警功能。

告警配置

默认项目

帮助文档

告警历史

告警策略

通知模板

新建

全部告警对象

请输入策略名称关键词

策略名称	描述	告警对象个数	告警频率	通知模板	运行状态	操作
告警测试	用于测试对应应用下...	1	每2小时告警一次	test通知模板01		告警历史 复制 删除

共 1 条

10 条 / 页

1 / 1 页

复制告警策略

- 进入 [数据连接器控制台](#) > [告警策略](#) 列表页。
- 找到需要复制的告警策略，单击操作列的复制。
- 在跳转页中修改所复制告警策略的相关信息，修改完后单击完成即可。

告警配置

默认项目

帮助文档

告警历史

告警策略

通知模板

新建

全部告警对象

请输入策略名称关键词

策略名称	描述	告警对象个数	告警频率	通知模板	运行状态	操作
告警测试	用于测试对应应用下的告警触发测试	1	每2小时告警一次	test通知模板01		告警历史  删除

共 1 条

10 条 / 页

1 / 1 页

删除告警策略

- 进入 [数据连接器控制台](#) > [告警策略](#) 列表页。
- 找到需要删除的告警策略，单击操作列的删除，在弹框中确认删除即可。

说明：
只有已禁用的策略才能被删除。

告警配置

默认项目

帮助文档

告警历史

告警策略

通知模板

新建

全部告警对象

请输入策略名称关键词

策略名称	描述	告警对象个数	告警频率	通知模板	运行状态	操作
告警测试	用于测试对应应用下...	1	每2小时告警一次	test通知模板01		告警历史 复制 

共 1 条

10 条 / 页

1 / 1 页

通知模板

通知模板

最近更新时间：2023-06-08 14:47:58

操作场景

本文指导您在数据连接器告警模块中通知模板相关内容。

支持多个策略一键复用模板，减少用户重复配置用户通知。可以自定义配置用户通知方式。例如：可以配置告警接收渠道为站内信、邮件、短信、微信、企业微信。并满足不同用户在不同通知时段生效。例如：A用户在白天接收告警通知；B用户在夜间接收告警通知。

通知模板支持新建、编辑、删除操作。

前提条件

- 查看通知模板：子账号需拥有数据连接器的项目读权限。
- 创建、编辑、删除通知模板：子账号需拥有数据连接器的项目写权限。

!

说明

详情可参见 [访问权限](#) 进行子账号授权。

使用限制

功能	限制
用户通知	最多可添加5项
接口回调	最多填写5个公网可访问的 URL

操作步骤

新建通知模板

- 登录数据连接器控制台，进入告警配置 > [通知模板](#) 页面。
- 单击新建，在“新建通知模板”中填写基本信息。
 - 模板名称：自定义模板名称。
- 填写通知操作，参数说明如下：
 - 用户通知

!

说明

- [系统管理员](#) 和 [项目管理员](#)：下拉支持选择当前项目下的所有成员账号。
- [项目成员](#)：只能选择该成员账号。
- [普通成员](#)：只能选择该成员账号。

参数	说明
接收对象	选择接收用户。
通知时段	定义接收告警的时间段。
通知周期	可按照天维度自定义通知周期，默认是每天通知。
接收渠道	支持站内信、邮箱、短信、微信、企业微信五种告警渠道。您还可以根据不同的用户维度，设置不同的告警接收渠道和通知时段，详情请参见 告警接收渠道 。

- 接口回调

参数	说明
接口 URL	填写公网可访问到的 URL 作为回调接口地址，最多可填写5个告警回调地址。数据连接器将及时把告警信息推送到该地址（例如：企

	业微信等）。
通知时段	定义接收告警的时间段。

- ❗ 说明：
- 回调地址保存成功后，当用户创建的告警策略被触发或被恢复均会通过接口回调推送告警消息。
 - 当用户创建的告警策略被触发或恢复时，均会通过接口回调推送告警消息。接口回调也支持重复告警。
 - 数据连接器告警配置的回调是通过 GET 方法调用回调地址，具体日志可通过您配置的服务地址的日志查看是否触发了回调。

4. 单击确定，即可新建一个通知模板。

← 新建通知模板

基本信息

模板名称 *

仅支持中英文、数字、下划线、-，最多25个字符

通知操作

用户通知

请确保所选接受对象已绑定手机，邮箱等通讯方式 [去查看](#)

接收对象

用户

请选择

添加用户

删除

通知周期

☒ 周一

☒ 周二

☒ 周三

☒ 周四

☒ 周五

☒ 周六

☒ 周日

通知时段

09:30:00 ~ 23:30:00

ⓘ

接收渠道

☒ 站内信

☒ 邮件

☒ 短信

☐ 微信 ⓘ

☐ 企业微信 ⓘ

+ 添加用户通知

接口回调 ⓘ

接口URL

删除

通知周期

☒ 周一

☒ 周二

☒ 周三

☒ 周四

☒ 周五

☒ 周六

☒ 周日

通知时段

09:30:00 ~ 23:30:00

ⓘ

+ 添加接口回调

确定

取消

修改通知模板

- 登录数据连接器控制台，进入告警配置 > 通知模板 页面。
- 找到需要修改的模板名称，单击名称进入编辑页面。

3. 进入编辑页面后，对需要修改的内容进行编辑，修改完后单击确定即可。

←

编辑通知模板

基本信息

模板名称 *

doc

通知操作

用户通知

接收对象

用户

test_alarm ×

新建用户

删除

通知周期

周一

周二

周三

周四

周五

周六

周日

通知时段

09:30:00 ~ 23:30:00

接收渠道

站内信

邮件

短信

微信

企业微信

+ 添加用户通知

接口回调

接口URL

https://qy...

删除

通知周期

周一

周二

周三

周四

周五

周六

周日

通知时段

09:30:00 ~ 23:30:00

+ 添加接口回调

确定

取消

删除通知模板

1. 登录数据连接器控制台，进入告警配置 > [通知模板](#) 页面。
2. 找到需要修改的模板名称，在其右侧的操作一栏中单击删除，在弹框中确认删除即可。

说明
被告警策略引用的模板无法删除，请先在告警策略中解除引用再进行删除。

告警历史 告警策略 通知模板				
新建 请输入模板名称关键词				
模板名称	用户通知对象	回调地址	关联告警策略	操作
dc			-	删除
test_			多个接口回调	删除

该通知模板已被引用在告警策略中，不允许删除

接口回调

最近更新时间：2023-06-08 14:47:58

操作场景

通过接口回调，您的企业微信群或自建系统可以直接收到腾讯云数据连接器监控的告警通知。接口回调具备将告警信息通过 GET 请求推送到可访问公网 URL 的功能，您可基于接口回调推送的告警信息做进一步的处理。

注意事项

- 当前告警回调没有认证机制，不支持 HTTP 认证。
- 当用户创建的告警策略被触发或恢复时，均会通过接口回调推送告警消息。接口回调也支持重复告警。
- 腾讯云数据连接器监控回调 API 出方向 IP 为动态随机分配，无法将具体的 IP 信息提供给您，建议您在安全组上配置加全放通策略。

操作步骤

1. 登录数据连接器控制台，进入告警配置 > [通知模板](#) 页面。
2. 单击新建，进入新建通知模板。
3. 在新建通知模板页配置完基础信息后，在接口回调模块中填写公网可访问到的 URL 作为回调接口地址（例如 `域名或IP[:端口][/path]` ），数据连接器将及时把告警信息推送到该地址。
4. 进入 [告警策略列表](#)，单击需要绑定告警回调的策略名称，进入管理告警策略页，并在告警策略页单击通知模板。

5. 告警消息将会通过 GET 请求推送到可访问公网 URL 地址。

基本信息

模板名称 *

test_01-2021-11-11_url_2

通知操作

用户通知

接收对象

用户

iPaas测试专用

新建用户

删除

通知周期

☒周一☒周二☒周三☒周四☒周五☒周六☒周日

通知时段

09:30:00 ~ 23:30:00

接收渠道

☒站内信☒邮件☒短信☐微信☐企业微信

+ 添加用户通知

接口回调

接口URL

https://520****01-2021-11-11.ipaas.beta.myqcloud.com/c53

删除

通知周期

☒周一☒周二☒周三☒周四☒周五☒周六☒周日

通知时段

09:30:00 ~ 23:30:00

接口URL

https://520****01-2021-11-11.ipaas.beta.myqcloud.com/c52

删除

通知周期

☒周一☒周二☒周三☒周四☒周五☒周六☒周日

通知时段

09:30:00 ~ 23:30:00

+ 添加接口回调

确定

取消

告警接收渠道

最近更新时间：2023-06-08 14:47:59

腾讯云数据连接器告警配置目前提供五种告警渠道，分别为：**站内信、短信、邮件、微信、企业微信**。
默认开启全部告警渠道，但开启所有告警渠道需在 [访问管理控制台](#) 填写验证接收人联系方式，包括手机、邮箱和绑定微信。

使用微信接收告警通知

配置微信告警渠道

用户使用微信接收告警信息前需于腾讯云控制台发起绑定接收人微信操作。指引如下：

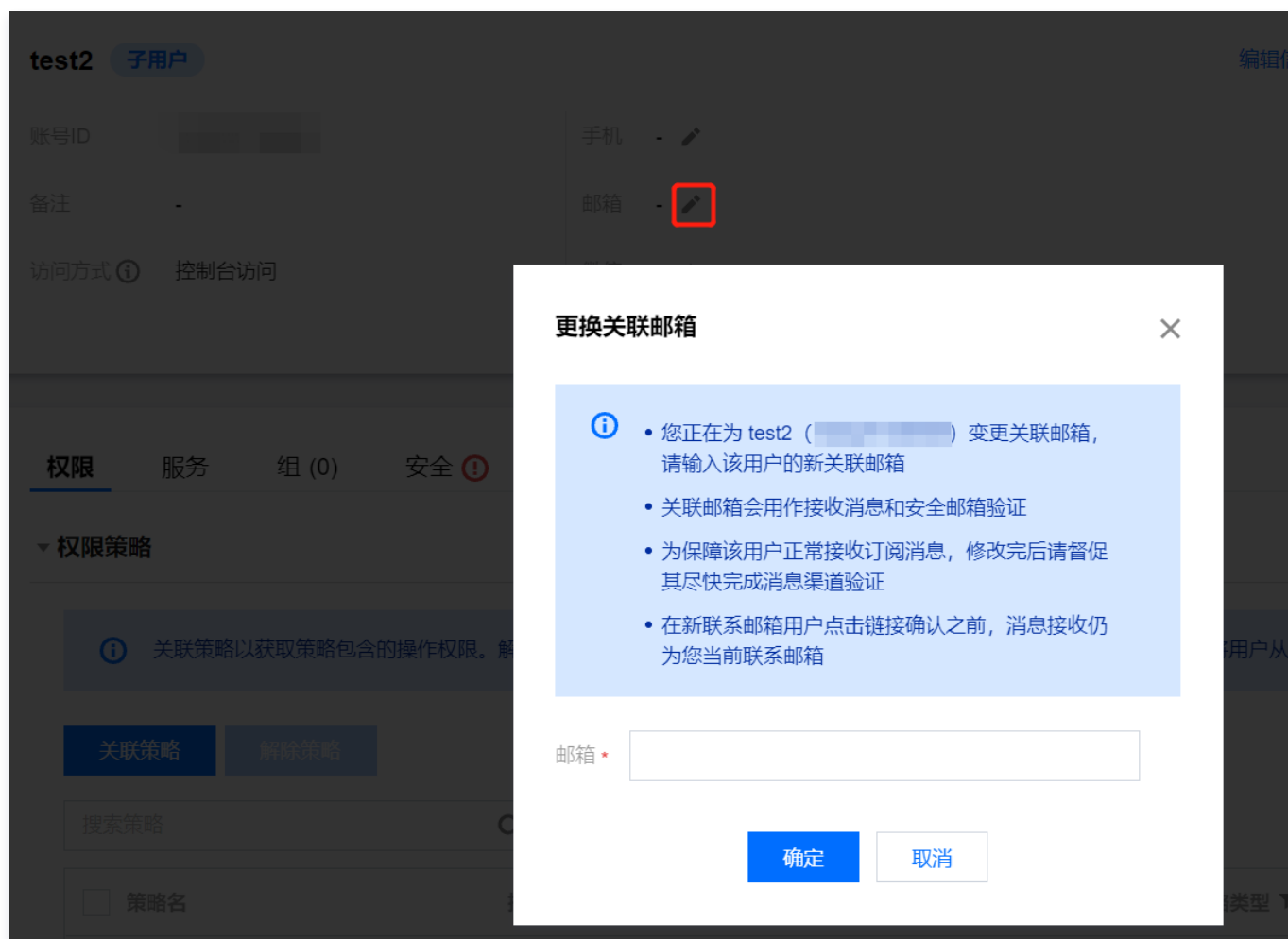
1. 登录 [访问管理控制台](#)。
2. 在左侧导航栏中，单击**用户** > [用户列表](#)，进入用户列表页面。
3. 找到需要配置接收微信通知的用户，单击用户名称进入用户详情页。
 - 3.1 填写并验证邮箱，用于接收微信验证消息（邮箱已验证的用户可忽略此步骤），请参见 [步骤1](#)。
 - 3.2 开启微信渠道并验证微信，请参见 [步骤2](#)。

步骤1：填写并验证邮箱

注意

此步骤用于接收微信验证消息，邮箱已验证的用户可忽略此步骤。

1. 如下图单击邮箱右边的编辑按钮，填写您的邮箱并单击**确定**。

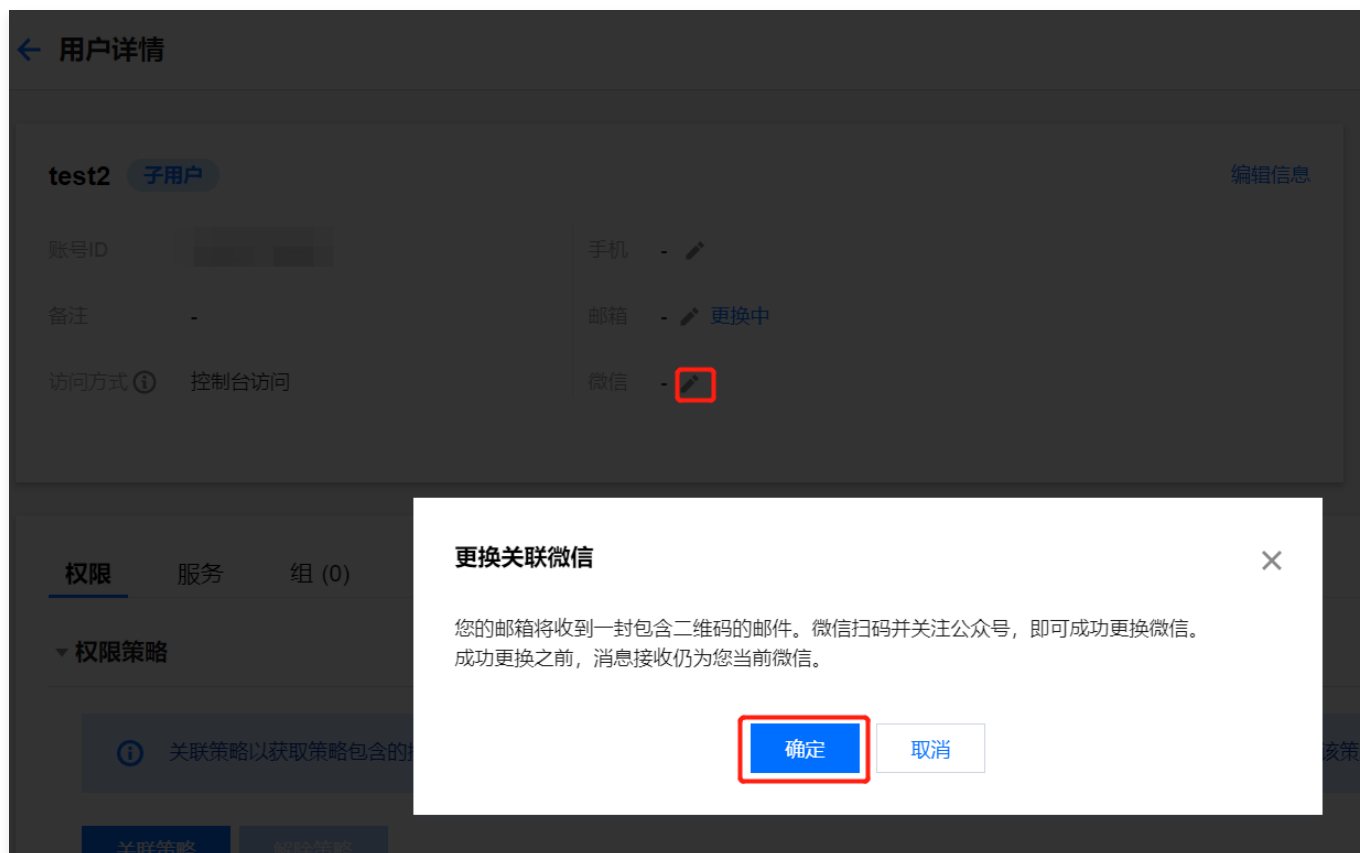


2. 登录您所填写的邮箱，在“腾讯云邮箱接收消息验证”邮件中，单击**确认接收**即可。



步骤2：开启微信渠道并验证微信

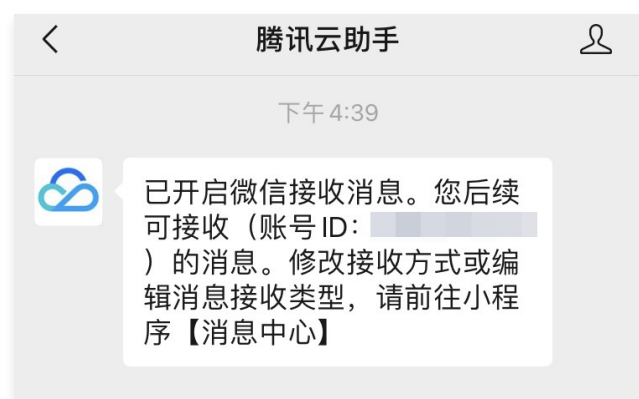
1. 返回用户详情页，单击微信右侧的**编辑**按钮，并在弹框中单击**确定**。



2. 在用户详情中单击 **更换中**，使用接收告警信息的微信客户端进行扫码。按照提示关注**腾讯云助手**服务号，即可完成绑定接收人微信。



3. 绑定成功如下图所示。



启用微信告警渠道

1. 进入 [腾讯云数据连接器控制台](#) > [告警配置](#) > [通知模板](#)。
2. 单击新建，进入新建通知模板。

3. 在新建通知模板页配置完基础信息后，在告警接收渠道勾选微信。

←

编辑通知模板

基本信息

模板名称 *

doc

通知操作

用户通知

接收对象

用户

iPaaS测试专用

新建用户

删除

通知周期

周一

周二

周三

周四

周五

周六

周日

通知时段

09:30:00 ~ 23:30:00

需开启微信通知才能接收公众号【腾讯云助手】的告警通知，[马上开启](#)

接收渠道

站内信

邮件

短信

微信

企业微信

+ 添加用户通知

版权所有：腾讯云计算（北京）有限责任公司

第216 共732页

4. 进入 [告警策略列表](#)，单击需要绑定微信告警的策略名称，进入管理告警策略页，并在告警策略页绑定通知模板。

← 编辑告警策略

基本信息

策略名称 *
doc

描述
请输入策略描述

告警规则

告警对象 *
应用 ⓘ
定时触发
+添加告警对象

发布环境
广州, 天津
删除

告警条件 *

指标
应用触发次数
+添加告警条件

判断逻辑
>
+添加告警条件

判断条件
1 次
删除

当以上 任意 告警条件, 按统计周期 1分钟 满足 持续一个周期 ⓘ 每5分钟告警一次

告警通知

通知模板 *
doc
+新建

确定

取消

使用短信接收告警通知

配置短信告警渠道

同微信和邮箱渠道类似，用户使用短信接收告警信息前需于腾讯云控制台发起绑定接收人短信操作。指引如下：

1. 登录 [访问管理控制台](#)。
2. 在左侧导航栏中，单击用户 > [用户列表](#)，进入 [用户列表](#) 页面。
3. 找到需要配置接收短信通知的用户，单击用户名称进入用户详情页。

4. 填写并验证手机号码，用于接收短信验证消息。

← 用户详情

test_alarm 子用户

编辑信息

账号ID 7XXXXXXXXXXXX

手机 -

备注 -

邮箱 -

访问方式 ⓘ 控制台访问

微信 -

标签 暂无标签

快速操作

订阅消息

删除用户

禁用用户

快捷登录

https://cloud.tencent.com/login/subAccount/7XXXXXXXXXXXX/Account&username=test_alarm

填写界面如下：

更换关联手机

ⓘ

- 您正在为 test_alarm (7XXXXXXXXXXXX) 变更关联手机，请输入该用户的新关联手机。
- 关联手机会用作接收消息和安全手机验证
- 为保障该用户正常接收订阅消息，修改完后请督促其尽快完成消息渠道验证
- 在新联系手机用户点击链接确认之前，消息接收仍为您当前联系手机

手机 *

+86

请输入手机号码

确定

取消

启用短信告警渠道

1. 进入 [腾讯云数据连接器控制台](#) > [告警配置](#) > [通知模板](#)。
2. 单击新建，进入新建通知模板。

3. 在新建通知模板页配置完基础信息后，在告警接收渠道勾选短信。

← 编辑通知模板

基本信息

模板名称 *
doc

通知操作

用户通知

接收对象
用户
iPaaS测试专用
新建用户
删除

通知周期
☒周一 ☒周二 ☒周三 ☒周四 ☒周五 ☒周六 ☒周日

通知时段
09:30:00 ~ 23:30:00

接收渠道
☒站内信 ☒邮件 ☒短信 ☒微信 ☒企业微信

+ 添加用户通知

接口回调

接口URL
https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=b1...
删除

通知周期
☒周一 ☒周二 ☒周三 ☒周四 ☒周五 ☒周六 ☒周日

通知时段
09:30:00 ~ 23:30:00

+ 添加接口回调

确定

取消

版权所有：腾讯云计算（北京）有限责任公司

第219 共732页

4. 进入 [告警策略列表](#)，单击需要绑定短信告警的策略名称，进入管理告警策略页，并在告警策略页绑定通知模板。

← 编辑告警策略

基本信息

策略名称 *
doc

描述
请输入策略描述

告警规则

告警对象 *
应用 ⓘ
定时触发
+添加告警对象

发布环境
广州, 天津
删除

告警条件 *

指标
应用触发次数
+添加告警条件

判断逻辑
>

判断条件
1 次
删除

当以上 任意 告警条件, 按统计周期 1分钟 满足 持续一个周期 ⓘ 每5分钟告警一次

告警通知

通知模板 *
doc
+新建

确定

取消

使用邮箱接收告警通知

配置邮件告警渠道

1. 前往 [用户列表](#) 页面。
2. 找到需要配置接收邮箱通知的用户，单击用户名称进入用户详情页。

版权所有：腾讯云计算（北京）有限责任公司

第220 共732页

3. 如下图单击邮箱右边的编辑按钮，填写您的邮箱并单击确定。



4. 在用户详情页邮箱右侧，单击发送链接验证。



5. 登录您所填写的邮箱，在“腾讯云邮箱接收消息验证”邮件中，单击确认接收即可。



启用邮件告警渠道

1. 进入 [腾讯云数据连接器控制台](#) > [告警配置](#) > [通知模板](#)。
2. 单击新建，进入新建通知模板。
3. 在新建通知模板页面配置完基础信息后，在告警接收渠道勾选邮件。

←

编辑通知模板

基本信息

模板名称 *

doc

通知操作

用户通知

接收对象

用户

iPaaS测试专用

新建用户

删除

通知周期

☒

周一

☒

周二

☒

周三

☒

周四

☒

周五

☒

周六

☒

周日

通知时段

09:30:00 ~ 23:30:00

接收渠道

☒

站内信

☒邮件

☒

短信

☒

微信

☒

企业微信

+ 添加用户通知

接口回调

接口URL

https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=

删除

通知周期

☒

周一

☒

周二

☒

周三

☒

周四

☒

周五

☒

周六

☒

周日

通知时段

09:30:00 ~ 23:30:00

+ 添加接口回调

确定

取消

版权所有：腾讯云计算（北京）有限责任公司

第222 共732页

4. 进入 [告警策略列表](#)，单击需要绑定短信告警的策略名称，进入管理告警策略页，并在告警策略页绑定通知模板。

← 编辑告警策略

基本信息

策略名称 *

doc

描述

请输入策略描述

告警规则

告警对象 *

应用 ①

发布环境

操作

定时触发

广州, 天津

删除

+添加告警对象

告警条件 *

指标

判断逻辑

判断条件

操作

应用触发次数

>

1

次

删除

+添加告警条件

当以上

任意

告警条件, 按统计周期

1分钟

满足

持续一个周期

!

每5分钟告警一次

告警通知

通知模板 *

doc

+新建

确定

取消

使用企业微信接收告警通知

配置企业微信告警渠道

步骤1：开通企业微信子用户管理

1. 进入 [访问管理](#)，单击左侧菜单栏的[联合账号](#) > [企业微信](#)，单击[立即开通](#)。
2. 在企业微信授权页，企业微信管理员使用企业微信或微信扫码登录，选择可见范围用户后单击[同意以上授权](#)，并添加。
3. 授权成功，被选的可见用户在企业微信上会接收到成功安装腾讯云助手的消息。

注意

非企业微信子用户或企业微信子用户但不在腾讯云助手可见范围内的用户无法通过企业微信接收消息。

步骤2：创建企业微信子用户

1. 单击左侧菜单栏[用户](#)>[用户列表](#)，单击[新建用户](#)。
2. 在新建用户页面，单击[企业微信导入](#)。
3. 选择企业名称，勾选用户，单击[下一步](#)，完成页面验证，企业微信子用户添加成功。

启用企业微信告警渠道

1. 进入 [腾讯云数据连接器控制台](#) > [告警配置](#) > [通知模板](#)。
2. 单击新建，进入新建通知模板。
3. 在新建通知模板页配置完基础信息后，在告警接收渠道勾选**企业微信**。

← 编辑通知模板

基本信息

模板名称 *
doc

通知操作

用户通知

接收对象

用户

IPaaS测试专用

新建用户

删除

通知周期

☒周一☒周二☒周三☒周四

通知时段

09:30:00 ~ 23:30:00

需开通企业微信子用户管理并且企业微信管理员授权成功才能接收【腾讯云助手】的告警通知，[马上开通](#)

接收渠道

☒站内信☒邮件☒短信☒微信*?*☒企业微信*?*

+ 添加用户通知

接口回调 *?*

接口URL

https://qyapi.weixin.qq.com/cgi-bin/webhook/send?key=b1

删除

通知周期

☒周一☒周二☒周三☒周四☒周五☒周六☒周日

通知时段

09:30:00 ~ 23:30:00

+ 添加接口回调

确定

取消

4. 进入 [告警策略列表](#)，单击需要绑定短信告警的策略名称，进入管理告警策略页，并在告警策略页绑定通知模板。

← 编辑告警策略

基本信息

策略名称 *

doc

描述

请输入策略描述

告警规则

告警对象 *

应用 ①

发布环境

操作

定时触发

广州, 天津

删除

+添加告警对象

告警条件 *

指标

判断逻辑

判断条件

操作

应用触发次数

>

1

次

删除

+添加告警条件

当以上

任意

告警条件, 按统计周期

1分钟

满足

持续一个周期

!

每5分钟告警一次

告警通知

通知模板 *

doc

+新建

确定

取消

告警历史

最近更新时间：2023-06-08 14:47:59

操作场景

腾讯云数据连接器告警配置提供查看告警历史功能，以便您回溯和查看近一个月的告警历史记录。您也可以在告警历史页快速查看告警策略。

操作步骤

查看告警历史

1. 登录腾讯云数据连接器控制台，进入告警配置 > 告警历史 页面。
2. （可选）单击左上角的“时间筛选”，筛选需要查看告警历史的时间范围。
支持近5分钟、近15分钟、近30分钟、近1小时、近6小时、近1天、昨天、近7天、近30天快速筛选，您也可以自定义时间范围。最多可查看近一个月的告警历史。
3. （可选）您可以在“告警对象”搜索框中输入告警对象的信息搜索对应的历史记录，例如：应用名称。
4. （可选）还可以根据 告警状态、策略名称、环境筛选出符合条件的告警历史记录。

告警历史							
告警策略 通知模板							
近7天	全部告警状态	全部告警策略	全部告警对象	天津	搜索	重置	
记录数量64条							
触发时间	告警状态	告警策略	告警对象	告警内容	持续时长	结束时间	操作
2022-09-02 03:17:00	持续中	流难度	监控告警测试	scheduler开头的流,集成流执行次数@总次数 >=1, 最新统计值: 3	7小时55分钟	-	查看详情
2022-09-01 16:59:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时14分钟	2022-09-01 17:13:00	查看详情
2022-09-01 16:29:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时27分钟	2022-09-01 16:56:00	查看详情
2022-09-01 15:57:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时5分钟	2022-09-01 16:02:00	查看详情
2022-09-01 15:47:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时4分钟	2022-09-01 15:51:00	查看详情
2022-09-01 14:28:00	已恢复	流难度	监控告警测试	集成流执行次数@总次数 >=1, 最新统计值: 0	12小时48分钟	2022-09-02 03:16:00	查看详情
2022-09-01 12:06:00	已恢复	流难度	监控告警测试	集成流执行次数@总次数 >=1, 最新统计值: 0	2小时21分钟	2022-09-01 14:27:00	查看详情
2022-09-01 08:01:00	持续中	流难度	监控告警测试	scheduler开头的流,集成流执行次数@总次数 >=1, 最新统计值: 3	0小时0分钟	-	查看详情
2022-09-01 07:10:00	已恢复	流难度	监控告警测试	集成流执行次数@总次数 >=1, 最新统计值: 0	0小时1分钟	2022-09-01 07:11:00	查看详情
2022-09-01 06:58:00	持续中	一二三四五一二三四五	复现问卷bug	应用触发次数 >=1, 最新统计值: 3	0小时0分钟	-	查看详情

重置筛选条件

您成功筛选告警历史信息后，在列表中单击重置即可。

告警历史

告警策略

通知模板

近7天

全部告警状态

doc

定时触发

天津

搜索

重置

记录数量5条

触发时间	告警状态	告警策略	告警对象	告警内容	持续时长	结束时间	操作
2022-09-01 16:59:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时14分钟	2022-09-01 17:13:00	查看详情
2022-09-01 16:29:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时27分钟	2022-09-01 16:56:00	查看详情
2022-09-01 15:57:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时5分钟	2022-09-01 16:02:00	查看详情
2022-09-01 15:47:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时4分钟	2022-09-01 15:51:00	查看详情
2022-08-30 16:16:00	已恢复	doc	定时触发	应用触发次数 >1, 最新统计值: 0	0小时8分钟	2022-08-30 16:24:00	查看详情

告警状态

告警状态	说明
持续中	没有被处理或正在被处理的告警
已恢复	已经恢复正常状态

集成工具

连接管理

最近更新时间：2024-08-22 14:34:21

操作场景

连接管理是当前账号下所有已授权应用账号的集合展示，同一应用可以添加多个连接配置，同一项目下的集成应用可以选择对应配置使用，也包含快速添加和修改操作，已使用的应用也会同步更新配置内容。

操作步骤

添加连接配置

步骤1：基本信息

登录 [腾讯云数据连接器控制台](#)，选择集成工具 > 连接管理，选择对应项目名称后单击添加连接配置进入连接配置添加界面。

连接管理

默认项目

帮助文档

添加连接配置

默认分组

请输入连接器名称

连接器名称	连接配置数	关联深度集成数	操作
企业微信机器人	2	2	展开
腾讯文档	2	0	展开
腾讯问卷	2	0	展开

步骤2：授权和配置

按照页面提示填写完成后单击下一步，进入授权/配置界面，不同的连接器对应第三方应用授权所需信息不同，请根据第三方获取密钥或账密文档后再填写。

新建连接配置

×

1 基本信息

>

2 授权/配置

连接器

维格表

✓

连接器版本

2.0.2

✓

连接器配置名称

维格表

✓

下一步

取消

步骤3：测试连接

新建连接配置

连接配置正确

✓ 基本信息

>

2 授权/配置

API token *

string

1. 登录维格表，点击左下角的个人头像，进入「用户中心」，前往「开发者配置」界面；2. 点击「+」号生成 API Token。注意：首次生成时需要绑定邮箱；3. 复制 API Token。

f(x)

维格表ID *

string

2. 打开任意一张维格表，在 URL 地址栏中找到以 dst 开头的一段字符串，即为这张维格表的ID

f(x)

服务商认证Token

string

3. 联系维格表的客服获取

f(x)

测试连接

上一步

完成

针对已存在的连接配置提供修改功能，但需要注意如下两点：

- 修改连接器配置，会影响到关联的集成应用。
- 若要修改连接配置生效，则需要对关联的集成应用任务进行重启，否则会导致运行中的集成应用任务运行异常，且重启会导致运行中集成应用任务的瞬时中断。

编辑连接配置

×

i

注意:

- 修改连接器配置，会影响到关联集成任务
- 修改连接配置要生效，则需要对关联集成任务及相关集成任务重新启动，否则会导致运行中的集成任务异常

▼ 需重启的关联任务 1 个

定时任务(企业微信服务 - 深度集成)

☐ 同意修改后进行任务重启

确定修改，并重启

取消

连接管理有分组概念，适用于不同环境不同账号的使用场景，切换分组可以快速切换到不同的配置进行连接。例如：正式环境使用分组一的配置，测试环境使用分组二的配置。

步骤1: 添加分组

可新建多个分组，每个分组对同个连接配置，配置不同的授权账号，对应不同的业务环境。

连接管理

企业微信服务

帮助文档

添加连接配置

同一个连接配置可在不同的分组中存在，不同分组的连接配置可以不同，便于快速切换，默认分组不可删除。

默认分组

默认分组

测试环境

正式环境

添加分组

请输入连接器名称

连接器名称	连接配置数	关联深度集成数
企业微信机器人	3	1
腾讯文档	1	0
腾讯问卷	2	0

步骤2：切换分组

在调试和发布时，可选择不同的分组调试或发布，方便用户一键切换调试或发布的环境。

定时任务

调试中

20211220_5.0

搜索流、组件

测试环境

退出调试模式

集成流

NewFlow

通用配置

自定义数据类型

连接器版本

企业微信机器人 2.0.0

自定义脚本

测试用例

TestSuite

TestSuite1

默认分组

测试环境

正式环境

关闭全部

Trigger

Scheduler scheduler

企业微信机器人 发送文本消息

安全网关

最近更新时间：2023-06-08 14:47:59

以下视频将为您介绍安全网关的相关概念及操作：

[观看视频](#)

简介

安全网关是为实现腾讯云数据连接器与用户内网服务集成互通而创新设计的代理系统，主要应用场景为用户内网服务不提供公网访问，而又希望通过腾讯云数据连接器（公有云部署）来完成与内网服务的集成。

安全网关由 Server 和 Agent 构成：

- Server 部署在腾讯云数据连接器系统内网，用户无需关注。
- Agent 部署在用户内网，允许根据地域或服务划分部署多个 Agent，通过安全网关转发来实现腾讯云数据连接器与用户内网服务的数据交互。

配置 Agent

步骤1：下载 Agent

1. 登录 [腾讯云数据连接器控制台](#)，选择**安全网关**。
 2. 单击**添加安全网关**，参见 [生成公私钥](#) 上传公钥。
 3. 配置 Agent IP 白名单和内网服务，单击**保存并确认**。
 4. 在安全网关列表中单击**下载 Agent** 按钮，下载 Agent。
- Agent 解压后目录结构如下：
 - bin 目录中包含 Agent 的可执行程序，按照不同的操作系统放在子目录 Linux、Windows、Mac 下。
 - configs 目录中包含 Agent 运行过程中所必须的配置，configs 目录介绍：
 - client 中存放 Agent tls 通信所必须的密钥等配置，与 Server 对应，此目录下文件**不可删除，不可修改**。
 - secret 目录下存放 Agent 连接 Server 时的私钥，私钥如何生成请参见 [生成公私钥](#)。
 - config.yaml 文件中包含 Agent 运行过程中必须依赖的配置。
 - logger_config.yaml 文件中包含 Agent 运行过程中的日志配置，**可修改日志级别及日志备份策略**。
 - log 中存放 Agent 运行过程中产生的日志。
 - scripts 目录存放 Agent 启动/停止脚本（ `start.sh/stop.sh` ）。

步骤2：Agent 日志配置

修改 Agent 文件夹 `ipaas-private-cloud-agent/configs` 目录下的 `logger_config.yaml` 文件，可以按需修改网关日志级别和日志备份策略等，`logger_config.yaml` 中各参数含义在文件中已做详细说明，本文不再介绍。

步骤3：启用 Agent

请按当前操作系统，执行以下相应启动脚本启动 Agent：

Mac 系统

执行以下命令启用 Agent：

```
./ipaas-private-cloud-agent/scripts/mac/start.sh
```

Linux 系统

执行以下命令启用 Agent：

```
./ipaas-private-cloud-agent/scripts/linux/start.sh
```

Windows 系统

执行以下命令启用 Agent:

```
./ipaas-private-cloud-agent/scripts/windows/start.bat
```

相关命令

各操作系统执行禁用 Agent 命令如下:

Mac 系统

执行以下命令可禁用 Agent:

```
./ipaas-private-cloud-agent/scripts/stop.sh
```

Linux 系统

执行以下命令可禁用 Agent:

```
./ipaas-private-cloud-agent/scripts/linux/stop.sh
```

Windows 系统

执行以下命令可禁用 Agent:

```
./ipaas-private-cloud-agent/scripts/windows/stop.bat
```

生成公私钥

步骤1: 检查 OpenSSL 版本

执行以下命令, 检查系统是否已安装 OpenSSL:

```
openssl version
```

执行命令如果能正常输出 OpenSSL 版本信息, 说明系统已安装 OpenSSL, 则可跳过下文步骤2: OpenSSL, 否则请参考下文步骤安装 OpenSSL。

步骤2: 安装 OpenSSL

MAC、Linux、Windows 系统安装 OpenSSL 方式不同, 详细安装步骤如下:

Mac 系统

执行以下命令安装 OpenSSL:

```
brew install openssl
```

Linux 系统

执行以下命令安装 OpenSSL：

Centos

```
yum install openssl
```

Ubuntu

```
sudo apt-get install openssl  
sudo apt-get install libssl-dev
```

Windows 系统

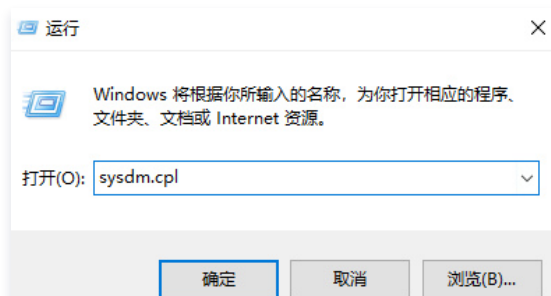
1. 下载 [OpenSSL](#) 安装包，选择不同的安装包下载。例如64位系统，则选择如图所示的安装包下载（选择 Light 的 EXE 版本即可）。

Download Win32/Win64 OpenSSL		
Download Win32/Win64 OpenSSL today using the links below!		
File	Type	Description
Win64 OpenSSL v1.1.1i Light EXE MSI	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1i (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.1i EXE MSI	63MB Installer	Installs Win64 OpenSSL v1.1.1i (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1i Light EXE MSI	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1i (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1i EXE MSI	54MB Installer	Installs Win32 OpenSSL v1.1.1i (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

2. 双击安装包，安装 OpenSSL。注意记录 OpenSSL 的安装目录（例如安装目录为 `C:\Program Files\OpenSSL-Win64`，后续 OpenSSL 配置环境变量时需要用到）。

3. 配置环境变量，以 Win10 系统为例：

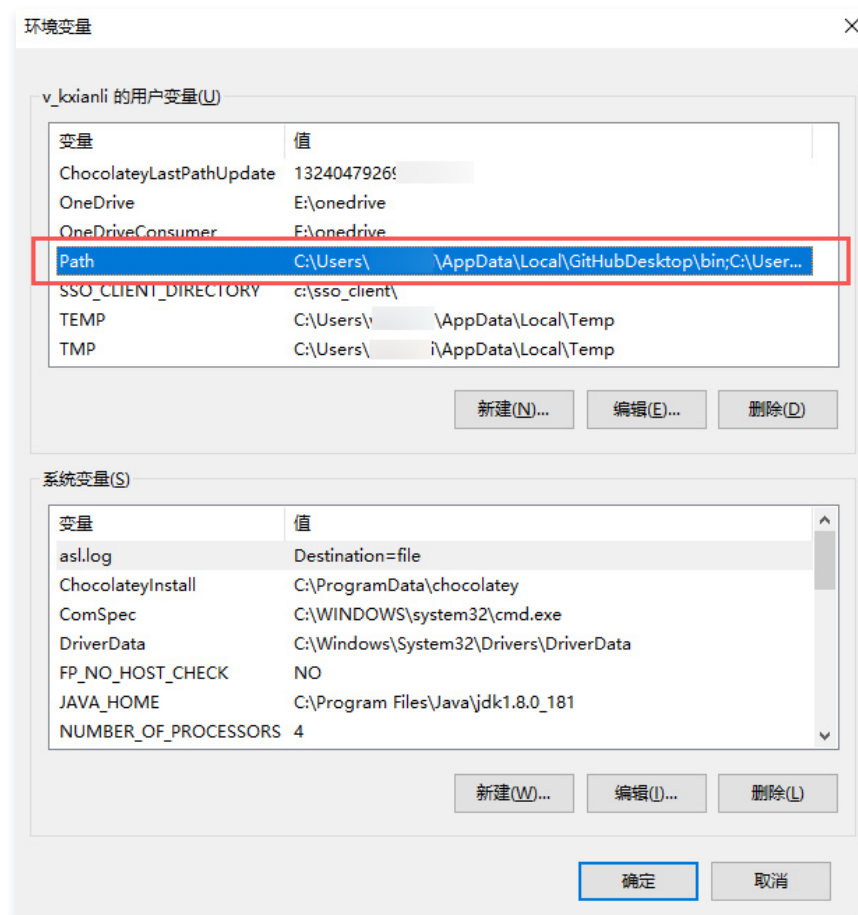
3.1 按 Win+R 键，在弹出的“运行”窗口中输入 sysdm.cpl，并按 Enter 键，打开“系统属性”窗口。



3.2 选择高级 > 环境变量进入环境变量页面。

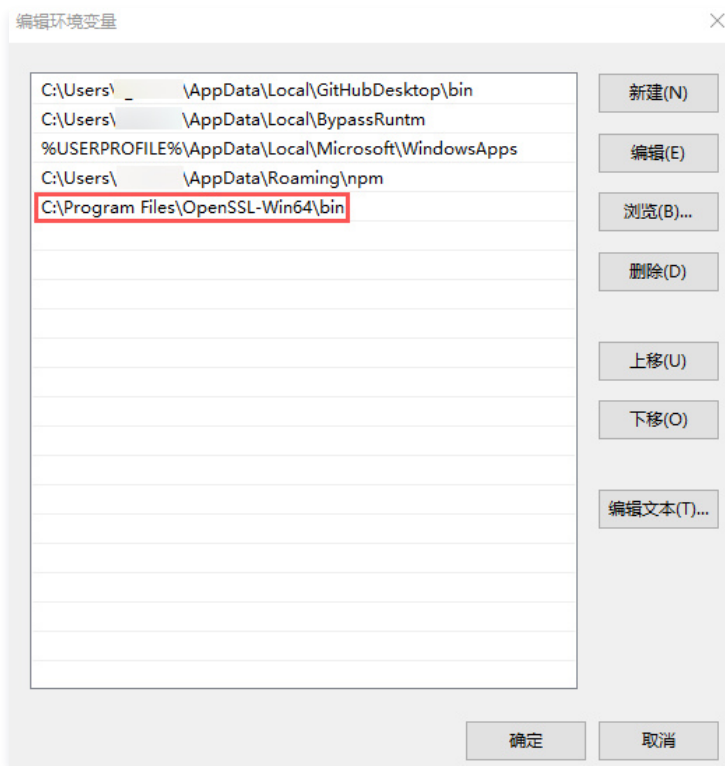


3.3 双击 Path 进入编辑环境变量页面。



3.4 单击新建，在左侧变量中输入 OpenSSL 安装目录的 bin 目录（bin 目录 = 安装目录 + bin，例如安装目录为

C:\Program Files\OpenSSL-Win64，则 bin 目录为 C:\Program Files\OpenSSL-Win64\bin）。



3.5 单击确定完成环境变量配置。

4. 安装验证。

4.1 按 Win+R 键，在弹出的“运行”窗口中输入 cmd，并按 Enter 键，打开“系统属性”窗口。

4.2 执行 openssl version 命令，如果出现 OpenSSL 版本信息，则说明 OpenSSL 安装成功，否则仔细核对 OpenSSL 安装步骤。

```
ca C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18362.1139]
(c) 2019 Microsoft Corporation. 保留所有权利。

C:\Users\>openssl version
OpenSSL 1.1.1i 8 Dec 2020

C:\Users\v_kxianli>
```

步骤3：生成及更新公私钥

1. 执行以下命令生成私钥。

```
openssl genrsa -out private.pem 1024
```

注意

请将生成的私钥放置在 ipaas-private-cloud-agent/configs/secret 目录下。

2. 执行以下命令，以上述私钥为基础生成公钥。当前目录下生成的 public.pem 即为公钥。

```
openssl rsa -in private.pem -RSAPublicKey_out -out public.pem
```

3. 生成新的私钥时，需替换 ipaas-private-cloud-agent/configs/secret 目录下的 private.pem 文件。

通用存储

最近更新时间：2023-09-19 10:33:21

使用场景

腾讯云数据连接器提供丰富的状态管理方案。登录 [腾讯云数据连接器控制台](#)，单击左侧工具栏集成工具 > 通用存储，进入“通用存储”功能，即可管理项目中使用到的存储结构和数据。

通用存储管理

“通用存储”主页面为存储管理页面。该页面下可以新建存储，同时查看当前已创建的所有存储列表，包括存储名、存储类型、存储内的数据量以及针对不同类型的存储可以执行的操作。

通用存储

默认项目

新增存储

沙盒

请输入存储名

存储名	存储类型	数据量（条）	操作
a	表结构	0	编辑 清空 更多
test	哈希	0	删除
test	表结构	0	编辑 清空 更多
test1	列表	0	删除
test2	哈希	0	删除
test3	哈希	0	删除
test4	列表	0	删除

新增存储

新增存储功能可以帮助您快速创建一个全新的存储类型。只需进行简单的配置即可完成创建工作、需要创建的存储名称以及对应的存储类型。

- 存储名称：请输入25位以内的英文字母或_。
- 存储类型：表结构、哈希结构、列表和字符串四种类型。

新增存储

×

存储名称

请输入存储名

请输入25位以内的英文字母或_

存储类型

表结构

确定

取消

表结构

编辑结构

用户可以使用该功能进行表结构的维护，包括列信息、索引信息、关系配置。

- 列信息：**表结构的列信息配置支持新增、修改和删除数据表中的列配置，可以快速完成结构编排。当前支持用户编辑列名称、列中数据的类型（支持 String、bool、int、float、decimal、datetime、date和时间）、值长度（当类型为 string 时）、精度值和小数位（当类型为 decimal 时）、是否为主键以及是否

支持自动编号（仅当该列为主键时）。

test表结构编辑

列信息 索引 关系配置

新建 删除

☐ 列名称	类型	值长度	精度值	小数位	设为主键	自动编号
☐ column2	string	64	--	--	☐	--
☐ column1	string	64	--	--	☐	--

保存 取消

- 索引配置：**索引配置用法同 MySQL 的 index 配置，用户可以通过此标签页建立索引，提高 MySQL 的检索速度。当前支持新建、删除索引、配置索引名称、配置索引类型（当前支持 index 和 unique 两种类型，具体区别请参考 [DOCUMENTATION](#)）、配置索引所在列（在已维护的列信息列表中选择）。

test表结构编辑

列信息 索引 关系配置

新建 删除

☐ 索引名称	索引类型	索引所在列
☐ index	index	column1

保存 取消

- 关系配置：**腾讯云数据连接器通用存储同时还支持用户配置外键信息。通过给外键命名、配置外键所在列、选择外键所对应的来源表和来源列信息即可完成配置。如果需要进行外键事件触发配置，则可以通过配置 on delete 和 on update 选项，请参见 [DOCUMENTATION](#)。

test表结构编辑

列信息

索引

关系配置

新建

删除

☐ 外键名	外键所在列	来源表	来源列	On Delete	On Update
☐ a	column1	a	asd	请选择	请选择
☐ b	column1	a	asd	请选择	Restrict

保存

取消

更多操作

- 清空表结构：“清空”操作允许用户清除当前表结构中的所有数据内容。



清空动作是高危操作，该操作将会清空本存储内的所有数据，且无法恢复，请谨慎使用。

通用存储

默认项目

新增存储

沙盒

请输入存储名

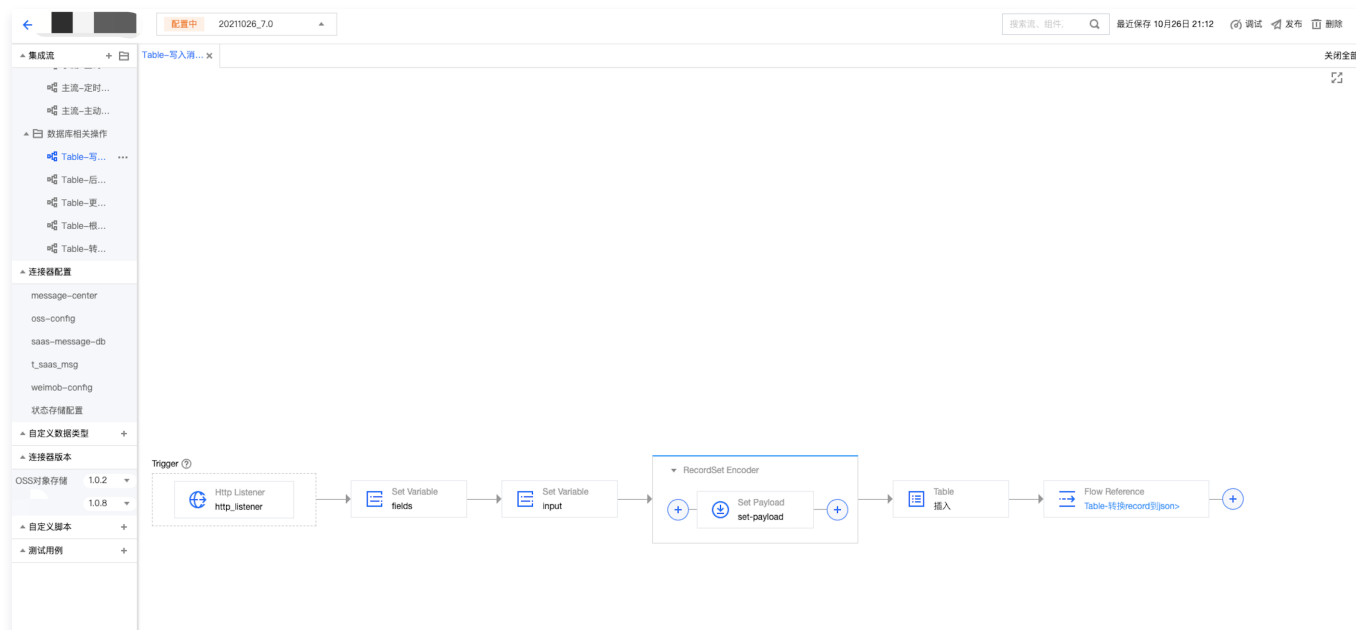
存储名	存储类型	数据量 (条)	操作
a	表结构	0	编辑 清空 更多
test	哈希	0	删除
test	表结构	0	编辑 清空 更多
test1	列表	0	删除
test2	哈希	0	删除
test3	哈希	0	删除
test4	列表	0	删除

- **复制同结构表：**通过“复制同结构表”功能可以基于现有的数据表结构快速创建一个新的数据表，方便用户进行数据结构的迁移和备份。

通用存储			
默认项目			
新增存储			
沙盒			
请输入存储名			
存储名	存储类型	数据量 (条)	操作
a	表结构	0	编辑 清空 更多
test	哈希	0	删除
test	表结构	0	编辑 清空 更多
test1	列表	0	删除
test2	哈希	0	删除
test3	哈希	0	删除
test4	列表	0	删除

- **删除：**“删除”操作可以删除本条存储记录。同样的，此操作被视为高危操作，会对线上数据造成影响，请确认后再进行。
- **使用示例：**通过集成应用向 table 插入数据。

1.1 创建应用，并在流中加入 table 组件节点。



1.2 配置 table 组件，填写想要添加的表名和表结构信息。

编辑连接器配置

Schema配置

表名 *
v_saas_msg

保存

1.3 完善集成应用的其他配置，发布并运行此应用以写入数据。

1.4 打开通用存储功能，观察到数据表结构已经成功生成。

通用存储

默认项目

新增存储

请输入存储名

存储名	存储类型	数据量 (条)	操作
	哈希	1	删除
.v_saas_msg	表结构	2	编辑 清空 更多

共 2 条

10 条 / 页

1 / 1 页

1.5 单击此存储名，可以查看通过集成应用写入的数据内容。

v_saas_msg

天津

基本信息

SQL 接口

新增

删除

导出

导入

msg_id	msg_business_id	msg_version	status	msg_event	msg_topic	retry_num	update_time	msg_public_account_k	state	msg_body	remark	create_time	mobile
cc4867a7-051c-4c8b-8f-1210083805	1210083805	1	0	orderStatusChange	ec_order	0	1630465332	100001710954	0	eyJkYXRIVGlZSi6MTYy		1630465332	
wwwwwwssdsd	1210083805	1	0	orderStatusChange	ec_order	0	1635252755	100001710954	0	eyJkYXRIVGlZSi6MTYy		1635252755	

共 2 条

10 条 / 页

1 / 1 页

哈希结构

哈希（hash）是根据关键码值（Key value）而直接进行访问的数据结构。它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。腾讯云数据连接器允许用户创建，哈希表结构：内容为 key-value 格式的数据。

列表

也可称为数组，是有序的元素序列，格式为一组拥有 ID 的数据信息。数组是用于储存多个相同类型数据的集合。

字符串

字符串（String）是由数字、字母、下划线组成的一串字符。腾讯云数据连接器通过通用存储功能，可以创建和使用字符串，帮助用户完成快速的复用，简化操作。

连接器

连接器市场

最近更新时间：2023-06-08 14:47:59

以下视频将为您介绍连接器的基本概念、功能及组成：

[观看视频](#)

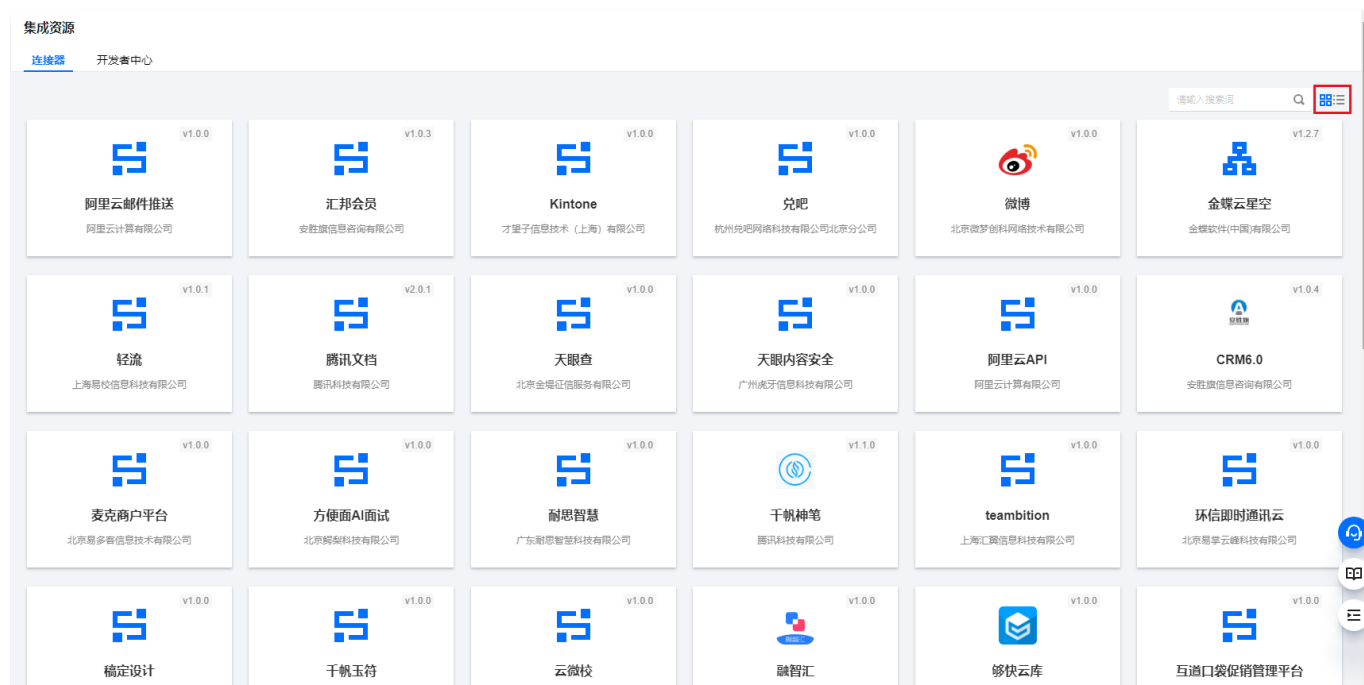
操作场景

集成资源连接器页面展示腾讯云数据连接器平台对外开箱即用的全部连接器资源，且部分连接器支持编辑，用户可根据业务需要进行对已有字段或者操作进行新增或删除。

操作步骤

查看连接器

登录 [腾讯云数据连接器控制台](#)，选择集成资源 > 连接器，连接器可根据您的喜好选择卡片或列表形式展示，默认是卡片形式展示，单击一个连接器卡片，进入到该连接器的详情界面，可查看该连接器的简介、更新日志、连接器文档等信息。



编辑连接器

部分连接器已开放编辑操作，您可登录 [腾讯云数据连接器控制台](#)，选择集成资源 > 连接器，单击选中一个连接器进入详情界面，单击编辑可以编辑此连接器。可进行重命名、修改连接器属性、增加或删除操作等操作，详情请参见 [我的连接器](#)。

← 连接器开放平台



企业微信

v1.0.8 ▾

组织： 腾讯科技有限公司 产品： 企业微信 发布者： 鹊桥 发布时间： 2021-12-07

编辑

连接器简介

集成企业微信成员管理和消息推送等API

更新日志

暂无

相关链接

[连接器文档链接](#)

[企业官网链接](#)

[产品官网链接](#)

连接器开发

最近更新时间：2023-06-08 14:47:59

以下视频将为您介绍连接器的常用封装方法、触发器开发及连接器调试开发：

连接器常用封装方法

[观看视频](#)

触发器开发

[观看视频](#)

连接器调试开发

[观看视频](#)

操作场景

集成资源设有开发者中心，支持用户按需自建连接器，我的连接器列表展示了当前账号下所有的自定义连接器，您可以浏览所有由您（主账号及子账号）创建、编辑、发布的连接器及其状态，便于管理和操作。

不同的连接器状态，对应“操作”下展示的功能是有区别的，具体如下：

状态	功能
已发布	允许用户进行查看、分享、编辑和删除
编辑中	允许用户分享、发布、编辑和删除
审核中	允许用户撤销、查看
已拒绝	允许用户编辑、删除
预发布	允许用户发布、编辑和删除

操作步骤

步骤1：添加连接器

当平台预提供的连接器不满足您的业务需求时，您可以在开发者中心添加新的连接器或者修改已有的连接器来方便您后续业务的使用。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击集成资源。

2. 在集成资源页面，单击 开发者中心 > 我的连接器 > 添加连接器。

集成资源

连接器

开发者中心

1

我的连接器

2

3

添加连接器

请输入搜索词

Q

连接器名称	连接器类型	连接器版本	创建时间	更新时间	状态	操作
易联云连接器	内部系统自建连接器	v1.0.0	2021-11-19 17:54:19	2021-11-19 17:54:19	预发布	编辑 删除
精采云连接器	内部系统自建连接器	v1.0.0	2021-11-19 17:54:18	2021-11-19 17:54:18	预发布	编辑 删除
腾讯文档	saas标准连接器	v1.0.5	2021-07-15 18:33:34	2021-11-18 16:13:48	编辑中	编辑 发布 删除
倍奇网	saas标准连接器	v1.0.1	2021-07-15 16:47:21	2021-11-18 16:13:30	编辑中	编辑 发布 删除

共 4 条

1

/ 1 页

3. 根据页面提示，配置信息后，单击确认进入编辑连接器界面进行配置。

步骤2：配置连接器

在配置连接器界面，您可以对连接器的基本信息、连接器属性、操作、触发器、自定义数据类型进行配置。

说明

您可以单击以下页签，查看具体的配置说明。

基本信息

在基本信息界面，您可以添加连接器的描述信息、上传连接器图标（不上传则使用默认图标 ），配置完成后单击保存。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

连接器ID *

tencent-docs

连接器显示名 *

腾讯文档

长度不超过16个字符

描述

tencent-docs

长度不超过500个字符

上传连接器图标（小图）

还未选择图片



选择图片

请上传png、jpg、svg格式，像素16*16并且100kb以内大小的图片

上传连接器图标（大图）

还未选择图片



选择图片

请上传png、jpg、svg格式，像素24*24并且100kb以内大小的图片

保存

重置

连接器属性

在连接器属性界面，您可以通过新建属性或编辑来新建或者修改属性，属性是连接器操作和触发器的一些共用参数。它们将会作为连接器的配置信息被使用。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

< 返回连接器属性列表

基本配置

属性名 *

test

属性名称只能包含字母、数字、中划线和下划线，只能以字母开头，长度不能超过50个字符

属性展示名

test

长度不超过16个字符

属性类型

string

默认值

请输入字符串

f(x)

必填

☐ 是否为必填

属性描述

选填

长度不超过200个字符

显示配置

属性展示分组

长度不超过50个字符

属性条件

属性选项卡

长度不超过20个字符

高级配置

表达式模式

支持表达式和普通输入

展示样例

长度不超过200个字符

是否为私有属性

☐ 是否为私有属性

密码模式

☐ 开启密码模式

内容校验

☐ 启用内容校验

保存

取消

单击配置测试连接，可以开启连接器的连接性测试功能，开通后后续您使用相关连接器时，填写完连接器信息后可以预先测试连接器的可用性。

测试连接功能

测试连接功能

☒ 开启测试连接功能 ⓘ

测试连接名 *

test

是否允许用户进行连接性测试

名称只能包含小写字母、数字、中划线，并且只能以小写字母开头，长度不超过50个字符

测试连接展示名

test

点击进入[流程预览](#)界面编辑测试连接器流程

确定

取消

单击流程预览，可以编辑测试连接器流程，您需要完成配置测试连接的具体实现。

连接器属性列表

触发器

表格数据同步

连接测试

操作

连接器配置

test

www



数据参考

:::	client_secret	string	第三方应用的密钥	编辑	删除
:::	code	string	发起授权成功后拿到的code	编辑	删除
:::	redirect_uri	string	授权回调地址	编辑	删除
:::	token	string	token	编辑	删除

单击连接器属性操作列的删除，可以删除此属性。删除后相关数据不可恢复，请您谨慎操作。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

新建属性

配置测试连接

请输入属性名

Q

连接器属性名	属性类型	属性展示名	①属性展示分组 ▾	操作
client_secret	string	第三方应用的密钥		编辑 删除
code	string	发起授权成功后拿到的code		编辑 删除
redirect_uri	string	授权回调地址		编辑 删除
token	string	token		编辑 删除
test	string	test		编辑 删除
test1	string	test1		编辑 删除

属性名称支持拖拽 调整顺序，列表中顺序将决定连接器属性在界面的展示顺序。详细展示效果请参考UI预览。

共 6 条

保存 重置

↺

🖨

☰

操作

当平台预提供的连接器操作不满足您的业务需求时，您可以修改已有连接器的操作或者是新增连接器并新增操作方便后续使用。在操作页面，您可以通过新建操作或者编辑对已存在的操作基本信息进行新建或编辑。操作封装了由集成应用主动发起的交互，在流中做流程编排和消息处理，并在处理完成后输出一个新的消息。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

新建操作

请输入操作名

Q

操作名	操作展示名	①操作展示分组 ▾	①私有操作 ▾	操作
do-request			是	编辑 删除 流程编辑
get-token-by-code			是	编辑 删除 流程编辑
get-token-by-refresh			是	编辑 删除 流程编辑
create-new-file	新建文档	文件操作	否	编辑 删除 流程编辑
query-the-doc	查询文档	文件操作	否	编辑 删除 流程编辑
updata-the-doc	更新文档	文件操作	否	编辑 删除 流程编辑
del-the-doc	删除文档	文件操作	否	编辑 删除 流程编辑
copy-the-doc	生成副本	文件操作	否	编辑 删除 流程编辑
add-folder	添加文件夹	文件夹操作	否	编辑 删除 流程编辑

↺

🖨

☰

所有的操作都需要对其具体的实现进行编辑。单击流程编辑，页面将跳转到画布中，并帮助您完成此次操作的具体实现，您可以像绘制一条集成流一样绘制此操作的流程。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

< 返回操作列表

操作基本信息

操作名 *

test

名称只能包含小写字母、数字、中划线，并且只能以小写字母开头，长度不超过50个字符

操作展示名

test

长度不超过16个字符

操作展示分组

操作描述

0

输出类型

☒ payload

☒ attributes

输出错误类型

☒ payload

☒ attributes

私有操作

☐ 是否为私有操作

操作参数配置

+ 新建操作参数配置

参数名称	参数展示名	参数展示分组	操作
------	-------	--------	----

参数名称支持拖拽，调整顺序，列表中顺序将决定其在界面的展示顺序。详细展示效果请参考UI预览。

流程编辑

开始流程编辑

保存

取消

单击具体属性的删除，可以删除此操作。删除后相关数据不可恢复，请您谨慎操作。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

预发布

UI预览

⋮	query-folder-permissions	查询文件夹权限	文件夹操作	否	编辑	删除	流程编辑
⋮	add-child-table	添加子表	表格操作	否	编辑	删除	流程编辑
⋮	clear-area-content	清空区域内容	表格操作	否	编辑	删除	流程编辑
⋮	del-rows-or-columns	删除多行（列）	表格操作	否	编辑	删除	流程编辑
⋮	view-doc-permissions	查看文档权限	权限操作	否	编辑	删除	流程编辑
⋮	set-doc-permissions	设置文档权限	权限操作	否	编辑	删除	流程编辑
⋮	sheet-get-getrows	获取表格指定行的内容	表格操作	否	编辑	删除	流程编辑
⋮	test	test		否	编辑	删除	流程编辑

操作名称支持拖拽 ⋮ 调整顺序，列表顺序将决定操作在界面的展示顺序。详细展示效果请参考UI预览。
共 18 条

保存 重置

触发器

当平台预提供的连接器的触发器不满足您的业务需求时，您可以修改已有连接器的触发器或者是新增连接器并新增触发器方便后续使用。在触发器页面，您可以通过新建触发器或者编辑对已存在的触发器基本信息及参数进行新建或编辑。触发器封装了由第三方发起、集成应用被动监听的交互，在流中作为源，在收到第三方系统的交互事件时在流中产生消息。触发器的使用流程与操作基本一致。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

预发布

UI预览

新建触发器

请输入触发器名

触发器名	触发器展示名	① 触发器展示分组	操作
⋮	doc-sync	表格数据同步	编辑 删除 流程编辑

触发器名称支持拖拽 ⋮ 调整顺序，列表中顺序将决定触发器在界面的展示顺序。详细展示效果请参考UI预览。
共 1 条

保存 重置

所有的触发器也需要对其具体的实现进行编辑。单击流程编辑，页面将跳转到画布中，并帮助您完成此次操作的具体实现。

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

[< 返回触发器列表](#)

触发器基本信息

触发器名 *

test1

✓

名称只能包含小写字母、数字、中划线，并且只能以小写字母开头，长度不超过50个字符

触发器展示名

test1

ⓘ

长度不超过16字符

触发器展示分组

ⓘ

长度不超过50字符

触发器描述

0

长度不超过500字符

输出类型

☒ payload

☒ attributes ⓘ

触发器参数配置

[+ 新建触发器参数配置](#)

参数名称	参数展示名	参数展示分组	操作

参数名称支持拖拽 ⓘ 调整顺序，列表中顺序将决定其在界面的展示顺序。详细展示效果请参考UI预览。

流程编辑

[点击开始编辑](#) ⓘ

保存

取消

单击具体触发器的删除，可以删除此触发器。删除后相关数据不可恢复，请您谨慎操作。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

新建触发器

请输入触发器名

触发器名	触发器展示名	① 触发器展示分组	操作
doc-sync	表格数据同步		编辑 删除 流程编辑

触发器名称支持拖拽，调整顺序，列表中顺序将决定触发器在界面的展示顺序。详细展示效果请参考UI预览。

共 1 条

保存

重置

预览

UI预览

自定义数据类型

在自定义数据类型页面，通过新建数据类型和编辑可以创建和编辑连接器对外交互的自定义数据格式。也可以通过上传 JSON 后台会自动识别数据类型并创建。需要注意的是，不同类型的数据结构类型对应的配置信息可能会有不同。

← 编辑连接器

配置连接器

测试连接器

编辑文档

发布连接器

基本信息

连接器属性

操作

触发器

自定义数据类型

< 返回数据类型列表

数据类型基本信息

数据类型名

类型名称只能包含字母、数字、中划线和下划线，只能以字母开头，长度不能超过50个字符

数据结构类型

struct

元素配置

+ 新建元素

元素名	元素展示名	元素类型	操作
-----	-------	------	----

元素名支持拖拽，调整顺序，列表顺序将决定其在界面的展示顺序。详细展示效果请参考UI预览。

共 0 条

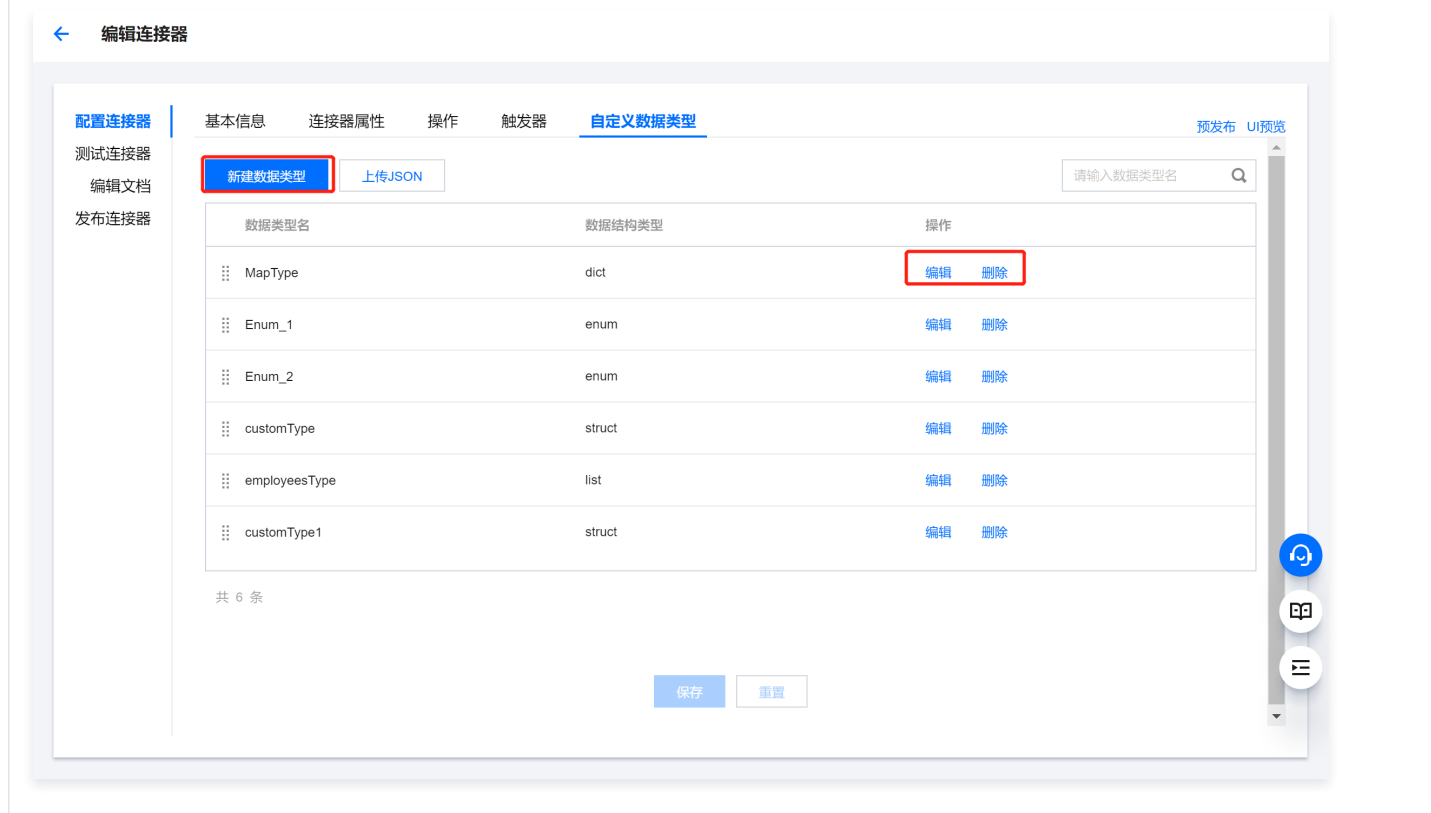
保存

取消

版权所有：腾讯云计算（北京）有限责任公司

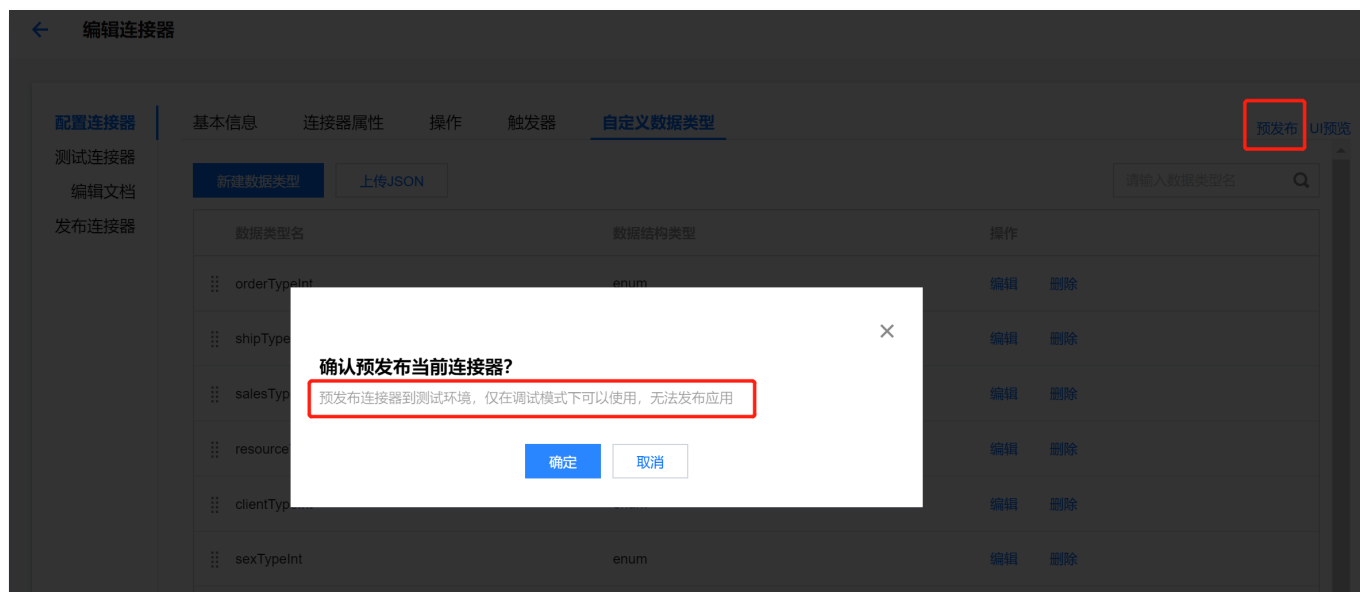
第252 共732页

单击具体数据类型的删除，可以删除此数据类型。需要注意的是：已经被引用的数据类型无法删除，需要先解绑定关系再进行删除操作。且删除后相关数据不可恢复，请您谨慎操作。



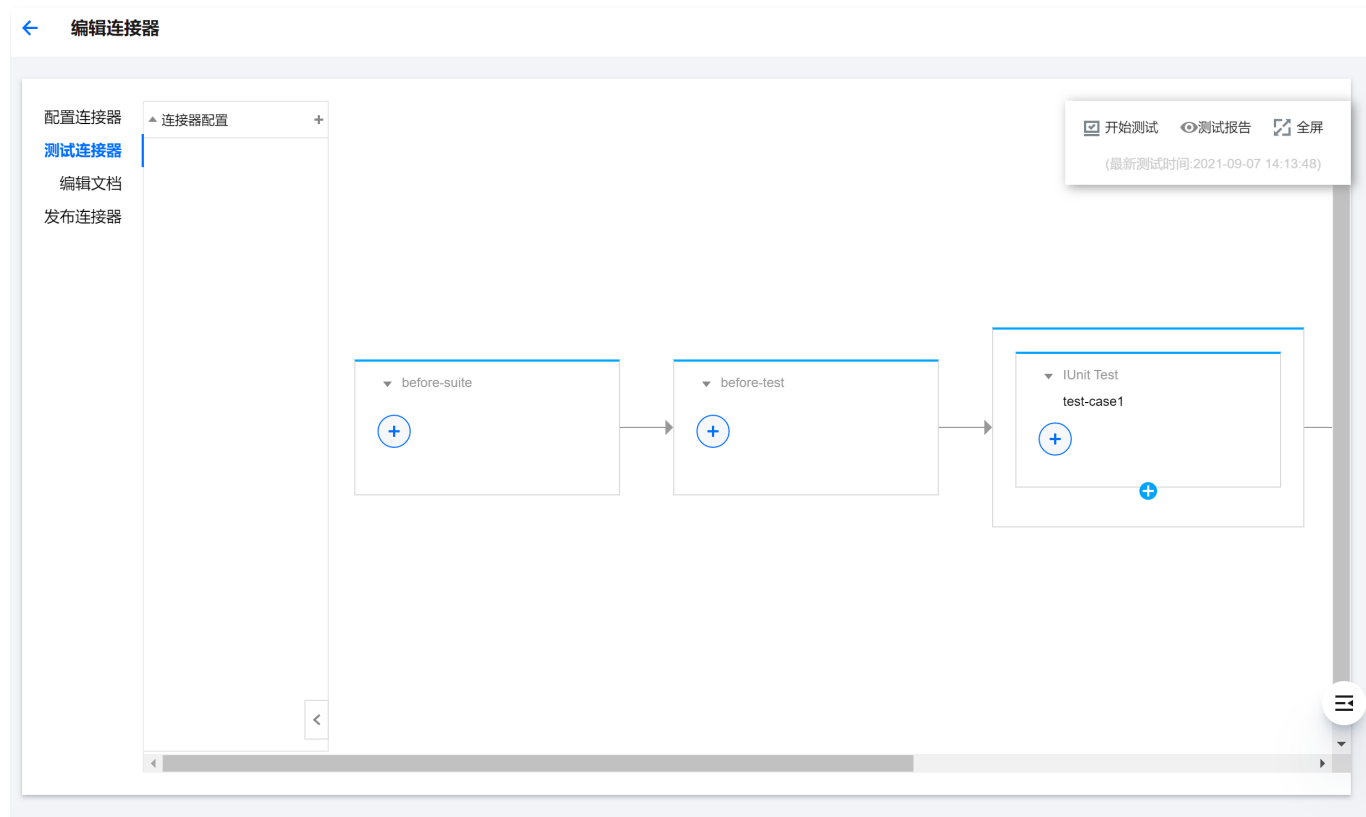
步骤3：预发布连接器

在配置连接器 > 自定义数据类型页面，单击编辑连接器右上角的预发布，可以发布连接器到测试环境。预发布之后的连接器可以在所有非正式环境的流中使用，同时允许用户对此连接器进行应用测试和单元测试，直到您觉得此连接器已经满足了所有的发布条件。需要注意的是，SaaS 类型的连接器将直接进入预发布流程，但是如果自建类型的连接器，您需要填写版本号之后才可以预发布成功。



步骤4：测试连接器

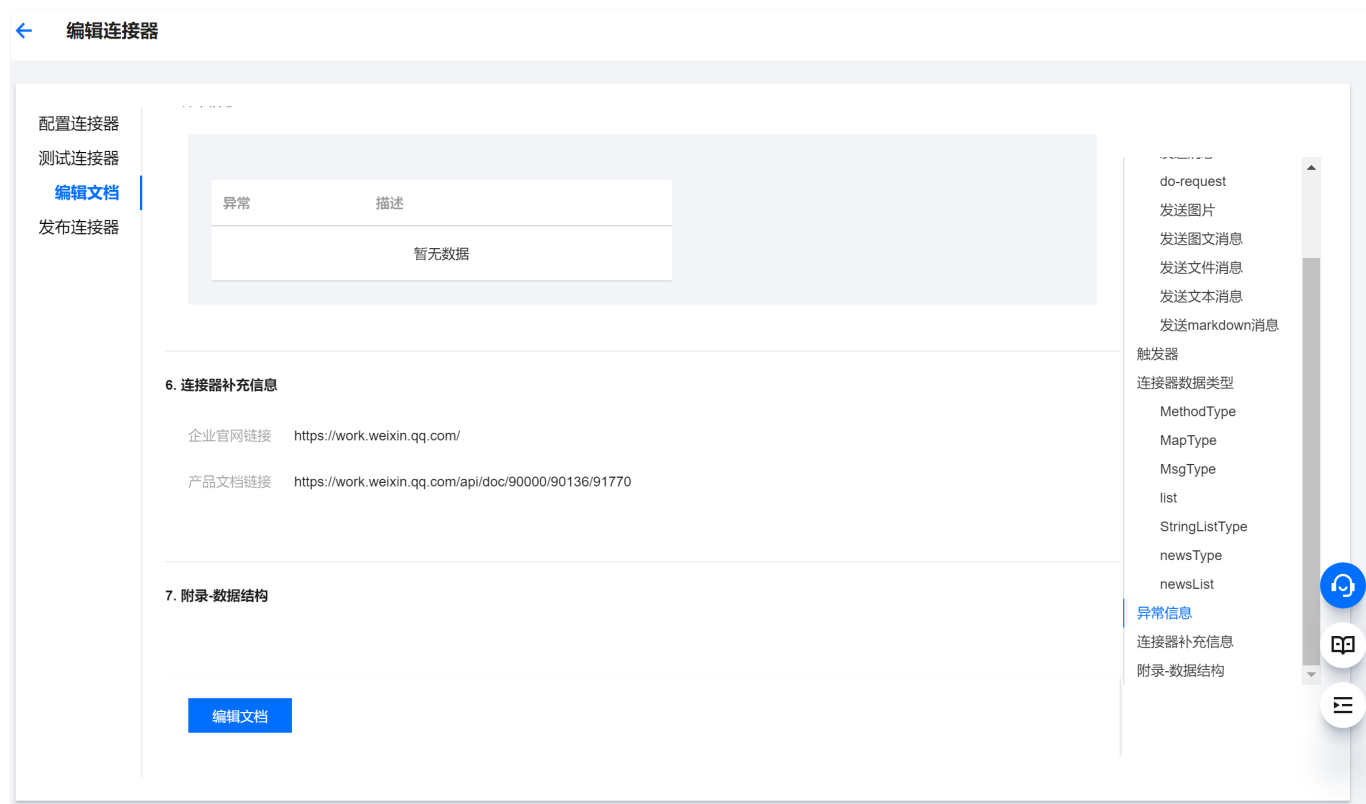
只有预发布成功的连接器才可以进行测试用例的配置，如果该连接器没有通过测试，则不能进行发布。测试连接器详细可参见 [应用配置 - 测试用例](#)。



步骤5：编辑文档

连接器文档中的大部分内容都会根据连接器的配置自动生成。您需要手动配置的内容如下，此类信息可以方便发布后其他用户进一步了解如何使用此连接器。

- 连接器操作/触发器是否配置返回值及返回信息
- 异常信息
- 连接器补充信息（例如官网链接和 API 文档地址等）



步骤6：发布/撤销连接器

在发布连接器之前，请确保连接器配置信息均已经完备和准确无误，一旦进入审核流程，您将无法直接修改此连接器的配置信息，直到审核流程结束或者您手动撤销审核。

在我的连接器界面单击**发布**或者在编辑连接器界面单击**发布连接器**，系统将进入连接器完成度校验，并且向您提供此次连接器发布审核的校验报告。如果存在错误，请参考提示文字并完善对应的配置内容，否则此连接器无法被发布；同时我们会帮助您标注出一些警告信息，警告信息并不会妨碍您连接器的正常发布，但是可能会对此连接器的使用造成一定的影响，建议您酌情修复。

← 编辑连接器

配置连接器
测试连接器
编辑文档
发布连接器

1 完整度校验 > 2 平台审核 > 3 上架成功

连接器信息存在一定风险，请留意。这些风险不影响提交审批流程，但是可能会影响审核的结果

❗ 错误：0项

⚠ 警告：13项

▲ 连接器配置

问题1	⚠ 操作send的参数msgType的描述为空
问题2	⚠ 操作send的参数content的描述为空
问题3	⚠ 操作do-request的参数method的描述为空
问题4	⚠ 操作do-request的参数path的描述为空
问题5	⚠ 操作do-request的参数queryParams的描述为空
问题6	⚠ 操作do-request的参数body的描述为空

▲ 测试信息

▲ 连接器文档

问题1	⚠ 操作send的返回值为空
问题2	⚠ 操作do-request的返回值为空
问题3	⚠ 操作send-image的返回值为空
问题4	⚠ 操作send-news的返回值为空
问题5	⚠ 操作send-file的返回值为空
问题6	⚠ 操作send-msg-text的返回值为空

提交安全合规审核

当一切准备妥当并且系统的自动完成度校验没有检测到问题的时候，您便可以提交此连接器进入人工审核的流程中。我们的工程师将会尽快审核您的变更申请，并且第一时间将审核结果反馈给您。您可以随时回到当前页面查看审核进度和结果，如果连接器审核通过，您将可以在连接器开放平台中看到它。若您还需要进一步修改，也

版权所有：腾讯云计算（北京）有限责任公司

第255 共732页

可以撤销发布。

集成资源

连接器 开发者中心

我的连接器

添加连接器 请输入搜索词

连接器名称	创建者	连接器类型	连接器版本	创建时间	更新时间	状态	操作
test1	大禹集团	saas标准连接器	无	2021-08-13 16:19:04	2021-08-13 16:20:18	编辑中	分享 编辑 发布 删除
test	大禹集团	saas标准连接器	无	2021-08-13 16:18:15	2021-08-13 16:18:15	编辑中	分享 编辑 发布 删除

共 2 条

1 / 1 页

相关操作

分享连接器

在我的连接器中添加的连接器，默认只能创建者和主账号能进行编辑。当有需要其它同事协作来完成连接器开发的情况下，可使用分享功能分享给主账号下的其它子账号，分享给其它子账号的成员将可对此连接器进行编辑。

- 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击集成资源 > 开发者中心 > 我的连接器。
- 在我的连接器页面，单击操作列的分享。

集成资源

连接器 开发者中心

我的连接器

添加连接器 请输入搜索词

连接器名称	创建者	连接器类型	连接器版本	创建时间	更新时间	状态	操作
test1	大禹集团	saas标准连接器	无	2021-08-13 16:19:04	2021-08-13 16:20:18	编辑中	分享 编辑 发布 删除
test	大禹集团	saas标准连接器	无	2021-08-13 16:18:15	2021-08-13 16:18:15	编辑中	分享 编辑 发布 删除

共 2 条

1 / 1 页

- 在分享弹框中，单击添加用户，选择当前主账号下的其他成员，单击确定即可。

分享

分享类型 指定用户可查看编辑

添加用户

用户名称	用户ID	操作
test_prj		删除
demo_user		删除

+添加用户

确定

取消

编辑连接器

- 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击集成资源 > 开发者中心 > 我的连接器。

2. 在我的连接器页面，单击操作列的编辑，前往编辑连接器界面。

集成资源

连接器 开发者中心

我的连接器 添加连接器 请输入搜索词

连接器名称	创建者	连接器类型	连接器版本	创建时间	更新时间	状态	操作
test1	测试账号	saas标准连接器	无	2021-08-13 16:19:04	2021-08-13 16:20:18	编辑中	分享 编辑 发布 删除
test	测试账号	saas标准连接器	无	2021-08-13 16:18:15	2021-08-13 16:18:15	编辑中	分享 编辑 发布 删除

共 2 条

1 / 1 页

3. 具体的编辑内容请参见 [配置连接器](#)。

连接器使用指南

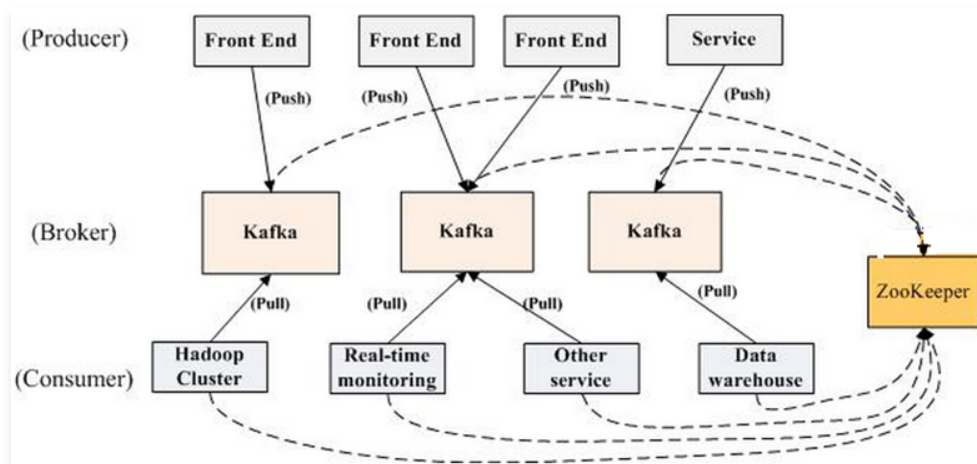
Apache Kafka 连接器

Apache Kafka 连接器使用指南

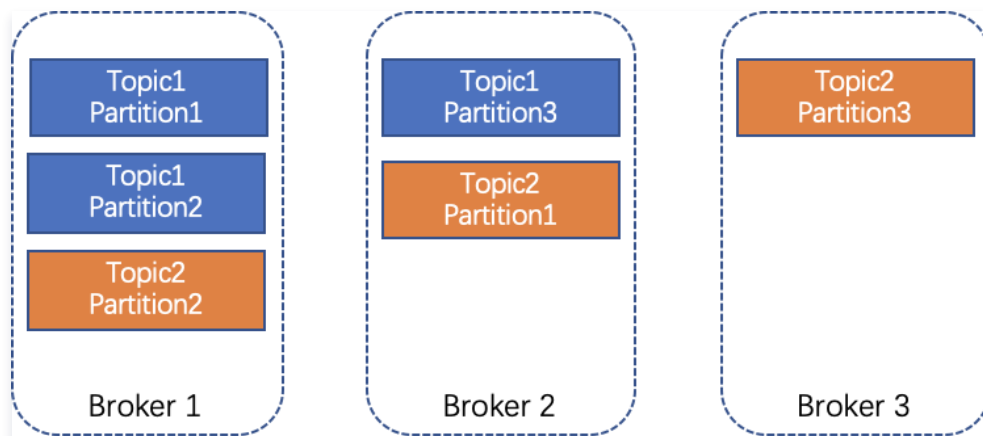
最近更新时间：2024-10-18 14:56:31

简介

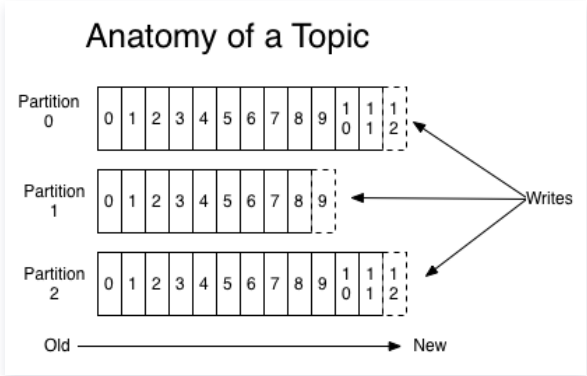
- Apache Kafka 连接器用于连接 Kafka 集群和腾讯云数据连接器集群，可消费 Kafka 集群的消息并作为集成流的 trigger，也可将集成流的消息生产发布到 Kafka 集群中。
- Apache Kafka 是一个分布式发布-订阅消息传递系统，kafka 有以下一些基本概念：
- Producer**：消息生产者，即向 kafka broker 发布消息的客户端。
- Consumer**：消息消费者，负责消费 Kafka 服务器上的消息。
- Topic**：主题，用于建立 Producer 和 Consumer 之间的订阅关系。
- Partition**：消息分区，一个 topic 可以分为多个 partition，每个 partition 是一个有序的队列，partition 中的每条消息都会被分配一个有序的 ID（offset）。
- Broker**：一台 kafka 服务器即为一个 broker，一个集群由多个 broker 组成，一个 broker 可以容纳多个 topic。
- Consumer Group**：消费者分组，用于归组同类消费者，每个 consumer 属于一个特定的 consumer group，多个消费者可以共同消费一个 Topic 下的消息，每个消费者消费其中的部分消息，这些消费者就组成一个分组，拥有同一个分组名称。
- Offset**：消息在 partition 中的偏移量，每一条消息在 partition 都有唯一的偏移量，消息者可以指定偏移量来指定要消费的消息。
- Producer、Consumer、Broker 关系如下图：



- Broker 保存消息数据，和 topic、partition 关系示意图如下：



- 同一 topic 的各 partition 逻辑关系如下图，kafka 仅保证同一 partition 下的消息有序。



连接器配置

通用配置标签页-连接配置

参数	数据类型	描述	是否必填	默认值
集群地址	string	Apache Kafka 集群地址，配置的格式：ip:port,ip:port	是	-
集群 Kafka 版本	enum	选择 Kafka 集群的版本号	否	1.1
SASL 安全认证模式	enum	选择连接到 Kafka 集群时的安全认证模式	否	PlainText
SASL 用户名	string	SASL/Plain 和 SASL/SCRAM 安全认证模式下的用户名	否	-
SASL 密码	string	SASL/Plain 和 SASL/SCRAM 安全认证模式下的密码	否	-
SASL/SCRAM 加密类型	enum	SASL/SCRAM 安全认证模式下的加密类型	否	-
使能 TLS 安全传输协议	bool	是否使用 TLS 加密和 Kafka 集群间的连接	否	false
TLS 客户端证书	file	可选，使用提供的证书对连接进行加密，仅当使能 TLS 安全传输协议设置为 True 才可配置	否	-
TLS 客户端Key	file	可选，使用提供的证书对连接进行加密，需和客户端证书同时提供，仅当使能 TLS 安全传输协议设置为 True 才可配置	否	-
TLS 服务端证书	file	可选，使用提供的证书对连接进行加密，仅当使能TLS安全传输协议设置为 True 才可配置	否	-

通用配置

连接配置

集群地址 *

string ▼

cl

f(x)

Apache Kafka 集群地址，配置的格式:ip:port,ip:port

集群Kafka版本

1.1 ▼

f(x)

SASL安全认证模式

SASL/SCRAM ▼

f(x)

SASL用户名

string ▼

f(x)

SASL密码

.....

f(x)

SASL/SCRAM加密类型

SHA512 ▼

f(x)

使能TLS安全传输协议

bool ▼

true ▼

f(x)

客户端证书文件内容

还未选择文件

选择文件

客户端key文件内容

还未选择文件

选择文件

服务端证书文件内容

还未选择文件

选择文件

通用配置标签页-消费者配置

参数	数据类型	描述	是否必填	默认值
消费者组	string	消费时的所属的消费者组名称	是	-
主题	string	订阅的主题列表，支持正则表达式	是	-
消费者数量	int	消费者组内的消费者数量，必须小于主题的 partition 数量	否	1

消费者配置

消费者组 *

string ▼

consumer1

f(x)

主题 *

string ▼

my-topic2

f(x)

订阅的主题列表，支持正则表达式

消费者数量

int ▼

1

f(x)

高级配置标签页-消费者组配置

参数	数据类型	描述	是否必填	默认值
消息确认模式	enum	消费消息后，在 Kafka 上确认 Offset 的模式，有两种： 消费后直到触发的流成功结束后才确认 Offset 消费后立即确认 Offset	否	流运行成功后确认
隔离级别	enum	Kafka 的事务隔离级别，可选 Read Uncommitted 和 Read committed	否	Read Uncommitted
重试阻塞时间	int	重试阻塞时间（毫秒），避免在失败场景下以紧密循环的方式重复发送请求	否	2000
请求超时时间	int	请求超时时间（毫秒）	否	30000
会话超时时间	int	会话超时时间（毫秒）	否	10000
心跳时间	int	心跳时间（毫秒）	否	3000
Rebalance 检测频率	int	Rebalance 检测频率（毫秒），Kafka 消费者组重平衡检测的间隔时间	否	60000
Rebalance 策略	enum	Rebalance 策略，按区间划分（Range）和轮询（RoundRobin）两种策略	否	Range
Rebalance 最大重试次数	int	Rebalance 最大重试次数	否	4
Rebalance 重试阻塞时间	int	Rebalance 重试阻塞时间（毫秒）	否	2000
最小拉取数据量	int	最小拉取数据量（字节）	否	1
默认拉取数据量	int	默认拉取数据量（字节），默认为 1048756(1024*1024)	否	1048576
最大拉取数据量	int	最大拉取数据量（兆），设置为0时无限制	否	0
拉取阻塞时间	int	拉取阻塞时间（毫秒）	否	250
提交频率	int	提交频率（毫秒）	否	1000
初始偏移量	enum	初始偏移量，取值为 Offset Newest 和 Offset Latest	否	Offset Newest
偏移量保留的时间	int	偏移量保留的时间（秒），0表示不过期	否	0

高级配置

消费者组配置

消息确认模式

流运行成功后确认

f(x)

隔离级别

Read Uncommitted

f(x)

重试阻塞时间
(毫秒)

int 2000

f(x)

避免在失败场景下以紧密循环的方式重复发送请求

请求超时时间
(毫秒)

int 30000

f(x)

会话超时时间
(毫秒)

int 10000

f(x)

心跳时间(毫秒)

int 3000

f(x)

Rebalance检测
频率(毫秒)

int 60000

f(x)

Rebalance策略

Range

f(x)

Rebalance最大
重试次数

int 4

f(x)

Rebalance重试
阻塞时间(毫秒)

int 2000

f(x)

最小拉取数据量
(字节)

int 1

f(x)

默认拉取数据量
(字节)

int 1048576

f(x)

最大拉取数据量
(兆)

int 0

f(x)

设置为0时无限制

拉取阻塞时间
(毫秒)

int 250

f(x)

提交频率(毫秒)

int 1000

f(x)

初始偏移量

Offset Newest

f(x)

偏移量保留的时间(秒)

int 0

f(x)

0表示不过期

高级配置标签页-生产者配置

参数	数据类型	描述	是否必填	默认值
最大消息限制	int	最大消息限制（字节），单条消息允许发送的最大数据量	否	1000000
确认模式	enum	确认模式，可选值为：None、LeaderOnly、All。其中 None 为不需确认，LeaderOnly 仅需 Leader 确认，All 需要 Leader 及所有 follower 都确认	否	LeaderOnly

请求超时时间	int	请求超时时间	否	10000
幂等性使能开关	bool	幂等性使能开关，设置为 true 时，打开 kafka 幂等性功能，实现 Exactly-Once 语义	否	false
消息刷新到磁盘的数据量	int	消息刷新到磁盘的数据量（字节），设置为0时不设置该参数	否	0
消息刷新到磁盘的消息数量	int	消息刷新到磁盘的消息数量，设置为0时不设置该参数	否	0
消息刷新到磁盘的间隔时间	int	消息刷新到磁盘的间隔时间（毫秒），设置为0时不设置该参数	否	0
消息刷新的最大消息量	int	消息刷新的最大消息量，设置为0时不设置该参数	否	0

生产者配置

最大消息限制
(字节)

int ▼1000000

f(x)

确认模式

LeaderOnly ▼

f(x)

请求超时时间
(ms)

int ▼10000

f(x)

幂等性使能开关

bool ▼false ▼

f(x)

打开kafka幂等性功能，实现Exactly-Once语义

消息刷新到磁盘
的数据量(字节)

int ▼0

f(x)

消息刷新到磁盘
的消息数量

int ▼0

f(x)

消息刷新到磁盘
的间隔时间(毫
秒)

int ▼0

f(x)

消息刷新的最大
消息量

int ▼0

f(x)

操作配置

Apache Kafka 连接器包含 Consumer 和 Publish 两种操作。

Consumer 操作

输入参数

None，Consumer 操作没有输入参数。

Apache Kafka ? Apache Kaf...

...

当前使用的是 kafka 物理配置

编辑 | 切换

配置

备注

输出

Consumer 操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 byte 数组
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	执行成功后，attribute 为 dict，属性列表见下表
variable	继承上个组件的 variable 信息

attrbites 属性列表：

attrbites 属性	数据类型	描述
topic	string	消息所属的 topic
partition	int32	消息的 partition
offset	int64	消息的 offset

例如：执行成功后，message payload 值为 byte 数组，message attribute 值如下：

```
{
  "topic": "test-topic",
  "partition": 1,
  "offset": 4
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "kafka: client has run out of available brokers to talk to (Is your cluster reachable?)"
}
```

案例

- 1. 按照 Kafka 集群信息，填写 kafka 连接器配置。

✓ 选择连接器

>

2 配置连接器

通用配置

连接配置

集群地址 *

string ▼

f(x)

Apache Kafka 集群地址，配置的格式:ip:port,ip:port

集群Kafka版本

1.1 ▼

f(x)

SASL安全认证模式

SASL/PlainText ▼

f(x)

SASL用户名

string ▼

f(x)

SASL密码

.....

f(x)

使能TLS安全传输协议

bool ▼

false ▼

f(x)

消费者配置

消费者组 *

string ▼

consumer1

f(x)

主题 *

string ▼

my-topic2

f(x)

订阅的主题列表，支持正则表达式

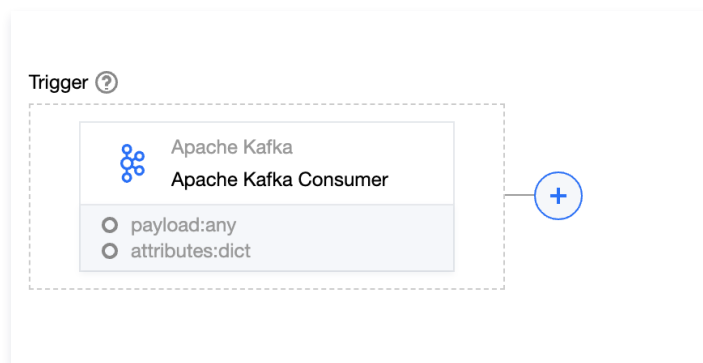
消费者数量

int ▼

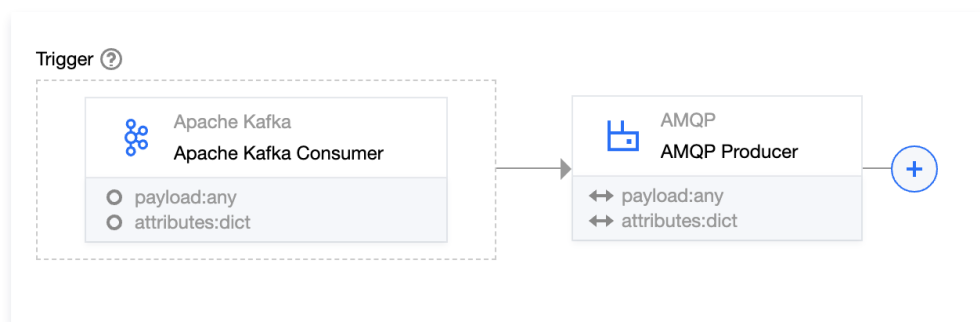
1

f(x)

2. 新建流，将 kafka consumer 作为 trigger 节点。



3. 添加其他业务逻辑，处理消费到的数据，例如：增加 amqp publish，将消费到的消息发布到 rabbitMQ。



4. 若配置正确，流发布后，可以看到 kafka 中的流被正确消费，且被发布到 rabbitMQ 中。

Publish 操作

输入参数

通用配置

参数	数据类型	说明	是否必填	默认值
要写入的 Kafka 主题名称	string	写入的 kafka 主题	是	—
要写入的 Kafka 消息内容	string或byte	待发布的 Kafka 消息内容	是	—

通用配置

要写入的 Kafka主题名称 *

string ▼

test-topic

f(x)

要写入的 kafka消息内容 *

string ▼

def dw_process(msg): return msg

🗑️

消息数据类型只能是string或者byte数组

高级配置

参数	数据类型	说明	是否必填	默认值
目标 partition	string	指定写入的 partition	否	—
key	string	key 相同的 message 会写入到同一 partition	否	—
header	string	header的 JSON 字符串	否	—

高级配置

目标partition

string ▼

请输入字符串

f(x)

Key

string ▼

请输入字符串

f(x)

key相同的message会写入同一partition

header

string ▼

请输入字符串

f(x)

输出

Publish 操作执行成功后，输出属性会保存在 message 消息体的 attribute；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	执行成功后，attribute 为 dict，属性列表见下
variable	继承上个组件的 variable 信息

attributes 属性列表：

attributes 属性	数据类型	描述
topic	string	消息所属的 topic
partition	int32	消息的 partition
offset	int64	消息的 offset

例如：执行成功后，message attribute 值如下：

```
{
  "topic": "test-topic",
  "partition": 1,
  "offset": 4
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "kafka: client has run out of available brokers to talk to (Is your cluster reachable?)"
}
```

案例

1. 按照 Kafka 集群信息，填写 kafka 连接器配置。

✓ 选择连接器

>

2 配置连接器

通用配置

连接配置

集群地址 *

string

f(x)

Apache Kafka 集群地址，配置的格式:ip:port,ip:port

集群Kafka版本

1.1

f(x)

SASL安全认证模式

SASL/PlainText

f(x)

SASL用户名

string

f(x)

SASL密码

.....

f(x)

使能TLS安全传输协议

bool

false

f(x)

消费者配置

消费者组 *

string

consumer1

f(x)

主题 *

string

my-topic2

f(x)

订阅的主题列表，支持正则表达式

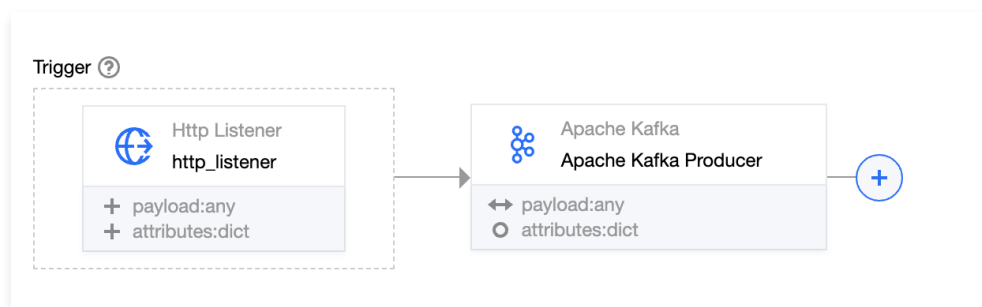
消费者数量

int

1

f(x)

2. 新建流，添加 trigger，例如：HTTP listener。



3. 在流中增加 kafka publish 操作，填写消息内容，将消息发布到 Kafka 集群中。

Apache Kafka

?

Apache Kaf...

...

☐

当前使用的是 kafka 物理配置

编辑 | 切换

配置

备注

通用

通用配置

要写入的 Kafka 主题名称

*

string ▼

test-topic

f(x)

要写入的 kafka 消息内容

*

string ▼

def dw_process(msg): return ms

🗑️

消息数据类型只能是 string 或者 byte 数组

高级配置

目标partition

string ▼

请输入字符串

f(x)

Key

string ▼

请输入字符串

f(x)

key 相同的 message 会写入同一 partition

header

string ▼

请输入字符串

f(x)

4. 发布流，触发流后，若连接参数及消息参数配置正确，消息将成功投递到所配置的 Kafka 集群的对应队列中。

阿里云 Kafka 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:00

操作场景

腾讯云数据连接器中的 Kafka 连接器支持 PLAINTEXT、SASL_PLAINTEXT、SASL_SCRAM 三种认证方式接入 Kafka 集群，可使用任一被支持的认证方式接入到 Kafka 集群。

前期准备

注意

在购买阿里云 Kafka 时，有 VPC 接入版和公网 + VPC 接入版可选，由于腾讯云数据连接器位于阿里云的外网，因此创建实例时请选择公网 + VPC 接入版。

步骤1：购买 Kafka 实例

1. 登录 [阿里云 Kafka 控制台](#)。
2. 单击**购买实例**，进入实例购买页面。
3. 按照实际付费方式选择包年包月或按量计费，单击**确定**进入购买页面（本文以按量计费为例）。

请选择您要创建的实例的付费方式

付费方式

包年包月

按量付费

请选择您要创建的实例的付费方式。点击“确定”后将进入购买页面。

确定

取消

4. 在实例购买页面，按照实际情况选择地域和规格类型。
实例类型请选择**公网 + VPC 实例**，按照实际情况选择公网流量大小，其他参数如磁盘类型、磁盘大小可参见 [阿里云 Kafka 文档](#) 说明，按照业务实际情况选择。

5. 勾选服务协议后，单击**立即购买**即可完成实例购买。

搜索...

[费用](#)
[工单](#)
[ICP 备案](#)
[企业](#)
[支持](#)
[App](#)
[📺](#)
[🔔](#)
[🛒](#)
[?](#)
[简体](#)

[前往购买页](#)

知而更尚规格，请提交[工单咨询](#)

磁盘类型

高效云盘

SSD

磁盘类型一经下单不可更改，请谨慎选择。
SSD 磁盘相比高效云盘拥有更高的 IOPS 性能，在消息高吞吐以及大量堆积等场景下性能更加稳定，推荐使用 SSD 磁盘类型。

磁盘容量

—

500

+

GB

数据默认 3 副本存储。
实例规格为标准版时，如购买 300G 磁盘，实际存储业务的磁盘大小为 100G，其余 200G 为备份容量。
实例规格为专业版时，如购买 300G 磁盘，实际存储业务的磁盘大小为 300G，**额外赠送 600G 备份容量**。

消息保留最长时间

—

72

+

小时

是指在磁盘容量充足的情况下，消息的最长保留时间；
在磁盘容量不足（即磁盘水位达到85%）时，将提前删除旧的消息，以确保服务可用性；
默认 72 小时，可选 24 小时 ~ 168 小时。

Topic规格

—

50

+

个

套餐内赠送一定数量的Topic（即最小值），这部分Topic不收费，如需额外增加Topic，按数量计费。
加购一个 Topic，会相应增加16个分区数。

规格达到上限时，由于底层优化，建议选择下一个流量规格，性价比更高。
专业版会附赠1倍的Topic数量，即实际可使用Topic数等于2倍的购买数，如果 Topic 数量多建议使用专业版。

购买数量

—

1

+

服务协议

☒ [消息队列 Kafka（按量后付费）服务协议](#)

总配置费用

¥

/小时?

立即购买

步骤2：创建资源

Kafka 实例部署

根据 [阿里云 Kafka 文档](#)，推荐在阿里云控制台手动创建规划好的 Topic 和 Consumer Group 等资源。

1. 登录 [阿里云 Kafka 控制台](#)，单击左侧菜单栏的**实例列表**，进入实例列表页面。
2. 单击操作列的**部署**，在弹出的部署页面中，关联已有 VPC 或新建 VPC。

消息队列Kafka版 / 实例列表

购买实例

自动续费

可按实例的名称和 ID 进行模糊搜索

Q

选择标签

名称	标签	状态	Topic	Group	规格类型	云监控	创建时间	操作
alikafka_post-cn- alikafka_post-cn-		● 未部署	0	0	标准版（高写版） 后付费 公网/VPC实例 高效云盘		今天 11:34:04	部署 更多

实例尚未部署，请点击右侧的“部署”按钮进行部署。

每页显示:

10

共1条

< 上一页

1

下一页

关联界面如下：

部署实例 alikafka_post-cn-

* VPC ID

请选择

您没有任何可用的 VPC。 [点击这里创建 VPC。](#)

* vSwitch ID

请选择

版本

0.10.2
2.2.0

* 消息保留时间

72

小时

* 最大消息大小

1

MB

* 消费位点保留时间

10080

分钟

自定义用户名密码

是
否

使用系统为每个实例分配的默认用户名和密码。

确定
取消

3. 单击页面中 VPC 创建链接，进入专有网络控制台，单击创建专有网络。



3.1 根据页面填写相关参数。

专有网络

交换机

路由表

NAT网关

公网NAT网关

VPC NAT网关

NAT网关资源包

终端节点

终端节点服务

网络智能服务

DHCP选项集

公网访问

弹性公网IP

Anycast弹性公网IP

共享带宽

共享流量包

IPv6网关

IPv6转换服务

公网质量工具箱

网间互联

VPN

云企业网

高速通道

智能接入网关

访问控制

网络ACL

← 创建专有网络

专有网络

地域

华南3（广州）

* 名称

ipaas-test

10/128

* IPv4网段

建议您使用RFC私网地址作为专有网络的网段如 10.0.0.0/8 , 172.16.0.0/12 , 192.168.0.0/16 。网段配置建议

10.0.0.0/8

一旦创建成功，网段不能修改

IPv6网段

不分配

描述

0/256

资源组

default resource group

交换机

创建交换机

删除

* 名称

ipaas-test

10/128

* 可用区

广州 可用区A

可用区资源

ECS

RDS

SLB

IPv4网段

网段配置建议

10

0

0

0

/

24

3.2 单击确定完成专有网络的创建。

专有网络

使用资源管理中心，可以帮助您从全局视角审视、管理所有云上资源。它包含按类型和地域查看云上资源，模糊搜索云上资源，常用实例快速查看和操作。您可以通过控制台顶部地域选择中查看各地域拥有的资源数（可能存在分钟级的数据延迟）。[开通资源管理](#)

路径分析功能 2022.1.24 迁移到NIS控制台；NIS控制台需要开通方可使用；子账号需获得 AliyunNISFullAccess 权限才可完整使用 NIS 功能。给大家带来不便还请谅解。

立即参与阿里云专有网络控制台易用性调研，有阿里云专属精美礼品等你拿哦。[点击进入](#)

创建专有网络

实例名称

请输入实例名称进行精确查询

标签筛选

下载

刷新

重置

☐	实例ID/名称	标签	网段	IPv6网段	状态	默认专有网络	路由表	交换机	云服务器	资源组	操作
☐	vpc- vpc-8m9kx2777h3gqgphs4k888 ipaas-test		10.0.0.0/8	开通IPv6	✓ 可用 未绑定云企业网	否	1	1	0	default resource group	添加云产品 删除
☐	设置标签										

4. 切换回阿里云 Kafka 实例部署页面，选中创建好的 VPC 和 vSwitch。

部署实例 alikafka_post-cn-qra2mgw4q01e

* VPC ID
vpc-vtb2b2a2777hucdphk4888

点击[这里](#)查看如何获取VPC信息。如有其它疑问，请提交工单。

* vSwitch ID
vsw-vtb2b2a2777hucdphk4888 可用区A

版本
0.10.2
2.2.0

* 消息保留时间
72
小时

* 最大消息大小
1
MB

* 消费位点保留时间
10080
分钟

自定义用户名密码
是
否

使用系统为每个实例分配的默认用户名和密码。

5. 单击确定后，等待实例部署完成。

消息队列Kafka版 / 实例列表

[购买实例](#)
[自动续费](#)

[选择标签](#)

名称	标签	状态	Topic	Group	规格类型	云监控	创建时间	操作
alikafka_post-cn- qra2mgw4q01e alikafka_post-cn- qra2mgw4q01e 部署中，预计需要10分钟~30分钟。请在部署成功后创建 Topic 和 Group。		部署中	0	0	标准版（高写版） 后付费 公网/VPC实例 高效云盘		今天 11:34:04	详情 更多

每页显示: 10 共1条 < 上一页 1 下一页

Kafka 资源创建

1. 在 [Kafka 控制台](#) 单击目标实例名称，进入实例管理页。

消息队列Kafka版 / 实例列表 / 实例详情

← alikafka_post-cn-qra2mgw4q01e
实例 ID alikafka_post-cn-qra2mgw4q01e 状态 ● 服务中 规格类型 标准版（高写版） 地域 华南3（广州）

[升配](#)
[转包年包月](#)
[降配](#)
[提交工单](#)
[释放](#)
[删除](#)

实例详情

Topic 管理

Group 管理

消息查询

监控报警

概览

当前实例没有任何 Topic

消息队列 Kafka 版里对消息进行的一级归类，例如可以创建 "Topic_Trade" 这一主题用来识别交易类消息。使用消息队列 Kafka 版的第一步就是先为您的应用创建 Topic。

创建 Topic

已用 Topic 数量

0 个 剩余可用 50 个

已用 Group 数量

0 个 剩余可用 100 个

已用分区数量

0 个 剩余可用 400 个

实例信息

云监控

实时统计

待调度预约的任务

任务执行记录

基础信息

实例 ID

alikafka_post-cn-qra2mgw4q01e

名称

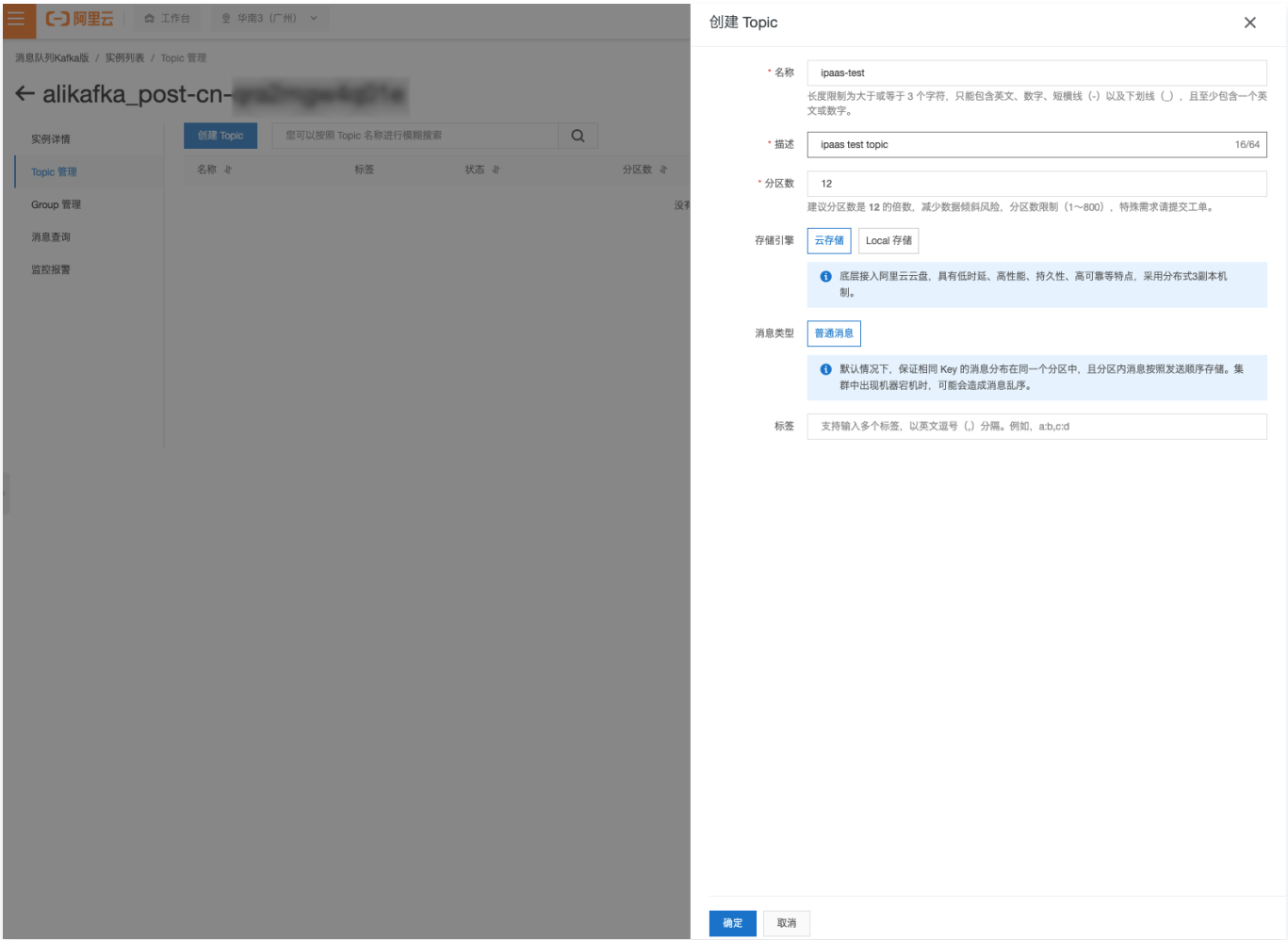
alikafka_post-cn-qra2mgw4q01e [编辑](#)

地域

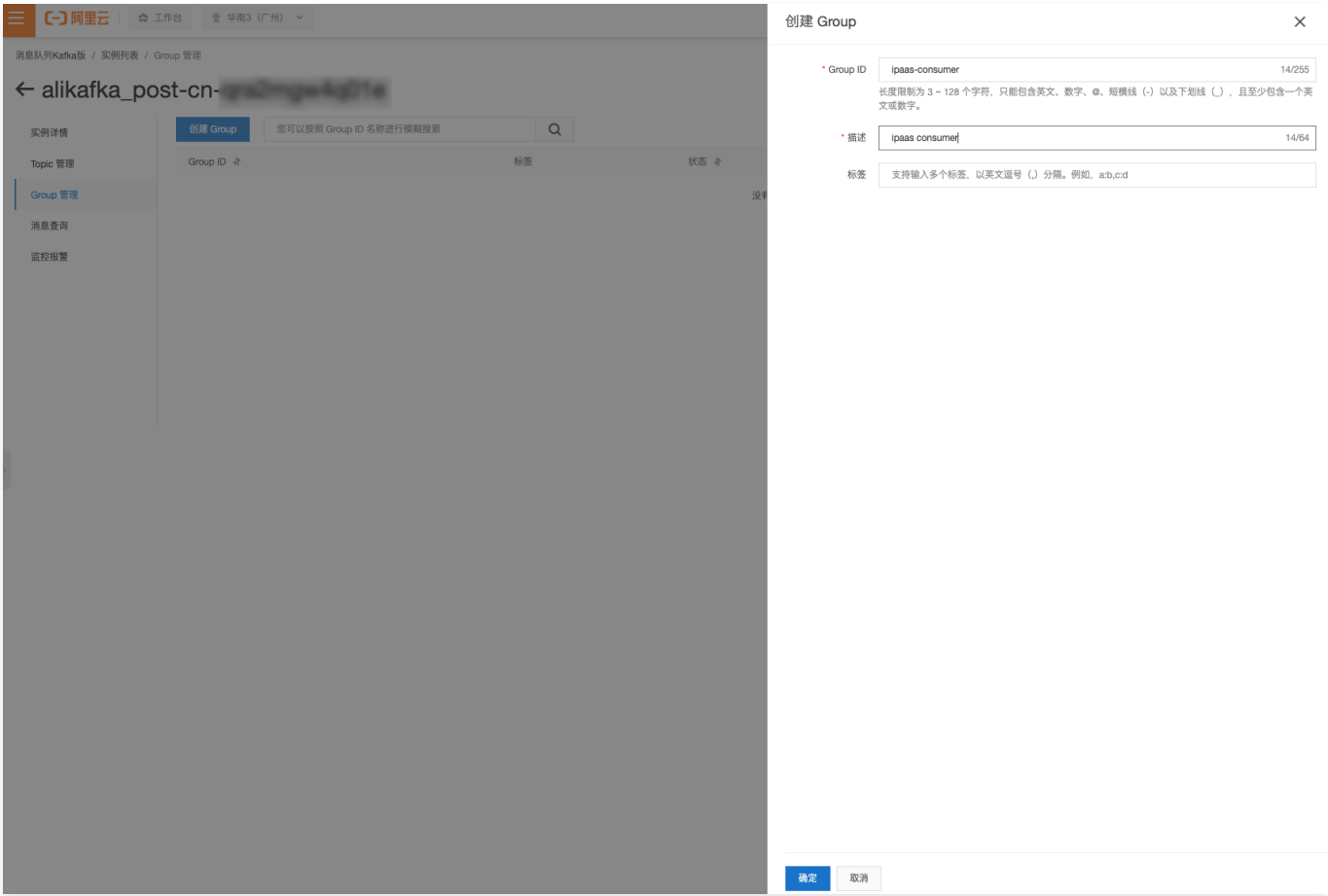
华南3（广州）

标签

2. 单击Topic 管理，单击创建 Topic，进入 Topic 创建页面，填写对应参数后，单击确定完成 Topic 创建。



3. 单击 **Group 管理**，单击**创建 Group**，进入 Group 创建页面，填写 group id 和描述等必填参数后，单击**确定**，完成 Group 创建。



接入配置

步骤1：获取 iPaaS Kafka 连接器配置信息

1. 登录 [阿里云 Kafka 控制台](#)，单击**目标实例 ID** 进入实例详情页，可以查询到接入点信息。

实例信息云监控实时统计待调度预约任务任务执行记录

基础信息

实例 ID	alikafka_post-cn- 	名称	alikafka_post-cn- 编辑
地域	华南3 (广州)	标签	
状态	● 服务中	规格类型	标准版 (高写版)
实例类型	公网/VPC实例	流量规格	alikafka.hw.2xlarge 读流量峰值 20 MB/s 写流量峰值 20 MB/s
公网流量	3 Mbps	磁盘类型	高效云盘
磁盘容量	500 GB	大版本	2.2.0
小版本	当前版本为最新版本		

接入点信息

SDK 中所配置接入点地址。点击[这里](#)了解具体使用方式。

类型	网络	协议	域名接入点	操作
默认接入点	VPC	PLAINTEXT	alikafka- -1-vpc.alikafka.aliyuncs.com:9092,alikafka-p- -vpc.alikafka.aliyuncs.com:9092	查看白名单 编辑白名单
SSL接入点	公网	SASL_SSL	alikafka- -j.alikafka.aliyuncs.com:9093,alikafka- -iikafka.aliyuncs.com:9093	查看白名单 编辑白名单

配置信息

2. 登录 [阿里云 Kafka 控制台](#)，单击目标实例 ID 进入实例详情页，下拉页面可以查询到配置信息。

配置信息			
VPC ID	vpc-	vSwitch ID	vsw-
可用区	可用区J	备用可用区	无
Topic 数量	50 个	Group 数量	100 个
分区数量	400 个	付费方式	后付费
创建时间	2022年4月13日 15:48:08	自由使用 Group	☐
消息保留时长	72 小时 编辑	最大消息大小	1 MB 编辑
消费位点保留时间	10,080 分钟 编辑	用户名	alikaafka_ 500m
密码	复制密码		

步骤2：配置腾讯云数据连接器 Kafka 连接器连接属性

1. 登录 [腾讯云数据连接器控制台](#)，单击新建应用，选择空白应用进行创建。

深度集成

默认项目

搜索应用、流、组件及配置

新手指引 帮助文档

应用管理

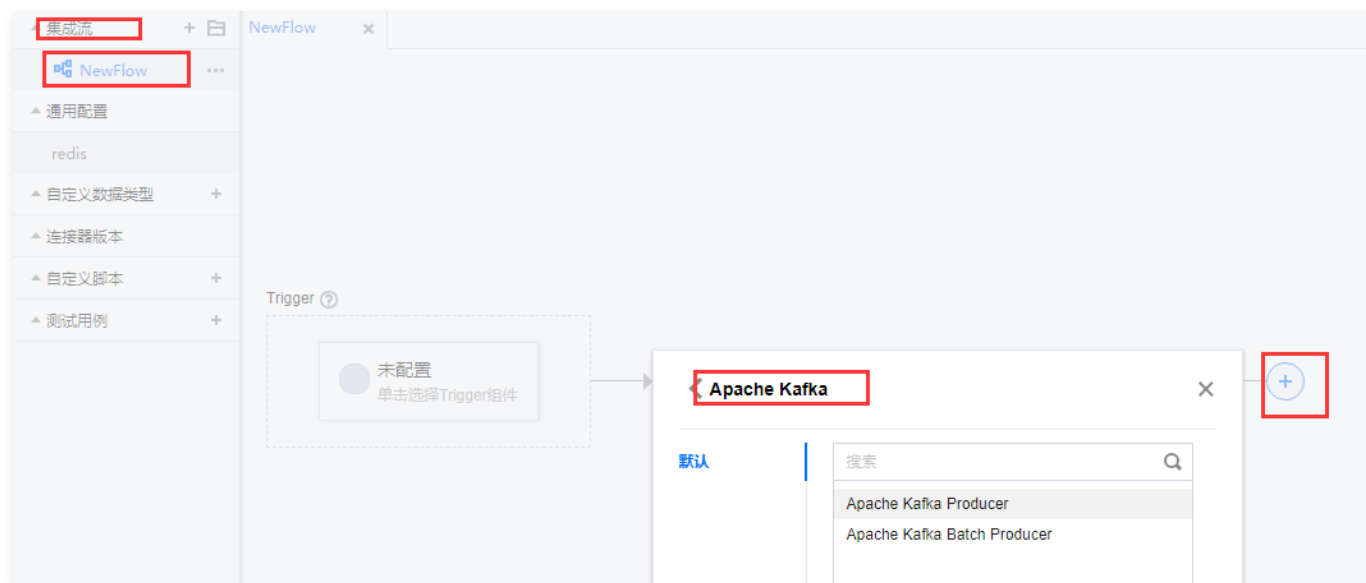
添加应用

导入应用

应用创建教学

应用名称	创建时间	更新时间	运行/调试版本	操作
问答信息	2022-04-12 17:18:56	2022-04-12 17:18:59	-	配置 修改名称 删除 导出
问答信息	2022-04-08 13:34:11	2022-04-08 13:35:27	运行中: 20220408_1.0(广州)	配置 修改名称 删除 导出 运行日志
问答信息	2022-03-30 19:31:29	2022-03-30 19:31:31	-	配置 修改名称 删除 导出

2. 单击左侧的集成流展开选择 NewFlow 在画布中单击 + 选择 Apache Kafka 连接器相关操作。



3. 单击右侧的**新建连接器配置**并按照指引填入相关参数。

新建连接器配置

1 选择连接器

2 配置连接器

连接器类型 *

Apache Kafka

配置名称 *

test

下一步

取消

4. 单击**下一步**，以使用 SASL/PlainText 接入点为例，在连接器配置参数中，依次填写如下配置：

新建连接配置

通用配置

连接配置

集群地址 *

string

alikaafka-...mz2ngek500m-1

f(x)

Apache Kafka 集群地址，配置的格式:ip:port,ip:port

集群Kafka版本

2.2

f(x)

SASL安全认证模式

SASL/PlainText

f(x)

SASL用户名

string

alikaafka_p...mz2ngek500m

f(x)

SASL密码

f(x)

使能TLS安全传输协议

bool

true

f(x)

客户端证书文件内容

还未选择文件

选择文件

f(x)

客户端key文件内容

还未选择文件

选择文件

f(x)

服务端证书文件内容

还未选择文件

选择文件

f(x)

其余字段按照默认数据填写即可，有其他需求可自行修改

- 集群地址：填入 Kafka 公网接入域名，此处填写 [接入点信息](#) 中网络为公网的域名接入点。
- 集群Kafka版本：根据购买的 Kafka 实例版本选择对应的版本，此处填写 [接入点信息](#) 中的大版本号。
- SASL安全认证模式：选择 SASL/PlainText。
- SASL用户名：此处填写 [配置信息](#) 中的用户名。
- SASL密码：此处填写 [配置信息](#) 中的密码。

- 使能TLS安全传输协议：选择 true。

消费者配置

消费者组 *

string ▾ ipaas-consumer

主题 *

string ▾ ipaas-test

f(x)

消费者数量

int ▾ 1

f(x)

订阅的主题列表，支持正则表达式

自行填写对应账户下的消费者配置，图中仅示例

- 消费者组：填写刚刚创建的 ipaas-consumer。
- 主题：填写刚刚创建的 ipaas-test。

5. 其他 Kafka 参数请按实际情况填写，也可参见 [Apache Kafka 连接器使用指南](#)。

华为云 Kafka 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:00

操作场景

腾讯云数据连接器中的 Kafka 连接器支持 PLAINTEXT、SASL_PLAINTEXT、SASL_SCRAM 三种认证方式接入 Kafka 集群，可使用任一被支持的认证方式接入到 Kafka 集群。

前期准备

前提条件

已配置好安全组规则和公网 IP，并开通 Kafka 服务。

华为云仅以下地域支持开通公网访问：

- 华北-北京一华北-北京四
- 华北-乌兰察布一
- 华东-上海一
- 华东-上海二
- 华南-广州

设置安全组

1. 登录华为云控制台，进入 [安全组](#) 设置。
2. 新建安全组或者修改已有安全组。



新建安全组

1. 单击控制台右上角的**创建安全组**，进入安全组创建页面，模板选择自定义。

创建安全组

★ 名称

sg-69bd

★ 模板

自定义

描述

入方向不放通任何端口，您可在安全组创建后，根据实际访问需求添加或修改安全组规则。

0/255

查看模板规则

确定

取消

2. 单击配置规则。

i 提示

安全组创建成功，请添加安全组规则以便能正常访问该安全组关联的实例。

配置规则

暂不配置

3. 添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	9094	0.0.0.0/0	通过公网访问 Kafka（关闭 SSL 加密）。
入方向	TCP	9092	0.0.0.0/0	通过 VPC 内网访问 Kafka（关闭 SSL 加密）。
入方向	TCP	9095	0.0.0.0/0	通过公网访问 Kafka（开启 SSL 加密）。
入方向	TCP	9093	0.0.0.0/0	通过 VPC 内网访问 Kafka（开启 SSL 加密）。
入方向	TCP	9999	0.0.0.0/0	访问 Kafka Manager。

修改已有安全组

1. 在安全组实例列表中，单击要修改的安全组实例右侧的**配置规则**，对已有规则进行修改。

安全组 

 使用指南 

 诚邀您参加虚拟私有云使用体验调研，您宝贵的意见和建议是我们持续提升产品体验的源动力，感谢您的参与！

名称 

名称	安全组规则	关联实例	描述	操作
sg-69bd 	4	0	入方向不放过任何端口，您可在安全组创建后，...	配置规则 管理实例 更多 
default	11	1	Default security group	配置规则 管理实例 克隆

2. 修改入方向规则，添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	9094	0.0.0.0/0	通过公网访问 Kafka（关闭 SSL 加密）。
入方向	TCP	9092	0.0.0.0/0	通过 VPC 内网访问 Kafka（关闭 SSL 加密）。
入方向	TCP	9095	0.0.0.0/0	通过公网访问 Kafka（开启 SSL 加密）。
入方向	TCP	9093	0.0.0.0/0	通过 VPC 内网访问 Kafka（开启 SSL 加密）。
入方向	TCP	9999	0.0.0.0/0	访问 Kafka Manager。

购买公网 IP

1. 进入 [弹性公网 IP 控制台](#)，单击右上角的**购买弹性公网 IP**。

弹性公网 IP 

 使用指南 

 诚邀您参加弹性公网 IP 使用体验调研，您宝贵的意见和建议是我们持续提升产品体验的源动力，感谢您的参与！

解绑

修改带宽

续费

更多 

所有状态 

弹性公网 IP 



标签搜索 







☐ 弹性公网 IP	监控	状态 	类型 	带宽 	带宽详情 	已绑定实例 	计费模式 	操作
 暂无表格数据								

2. 按实际情况选择计费模式，设置区域，线路和带宽等参数。

3. 单击立即购买。

购买弹性公网IP

放心购 灵活调整

计费模式

包年/包月 按需计费

区域

华南-广州

弹性公网IP仅支持绑定在处于相同区域的云资源上。购买后不能更换区域，请谨慎选择。

线路

全动态BGP 静态BGP

不低于99.95%可用性保障

带宽大小

1 2 5 10 100 200 自定义

带宽范围: 1-2,000 Mbit/s

IPv6转换

一键开启, 实现对外提供IPv6访问能力

公测期间IPv6转换功能免费。

弹性公网IP名称

高级配置

带宽名称 标签

监控

默认开启基础监控

免费提供分钟级粒度的流量监控 监控带宽流量波动、出入网带宽速率等指标详情

购买时长

1 2 3 4 5 6 7 8 9个月 1年 2年 3年 4年 5年

自动续费

弹性公网IP费用: 免费 + 带宽费用: ¥

立即购买

接入配置

步骤1: 设置 Kafka 接入方式

1. 登录 消息队列 Kafka 控制台，在实例列表中选择要设置的实例。

Kafka专享版

使用指南 购买Kafka实例 免费领取Kafka实例

HOT 【日常任务】前往会员中心领取日常任务，赢码豆换好礼!

快速入口: 购买RabbitMQ专享版 购买RocketMQ实例

重启 转包周期 续费 更多

名称 监控 状态 版本 规格 已用/可用存储空间 (GB) 分区上限 (个) 计费模式 操作

运行中

2.3.0

1u2g.single

3/65

10

包年/包月 181天后到期

重启 查看连接地址 更多

2. 单击实例名称进入详情页，单击安全组，完成安全组绑定。
若前置条件中直接修改的是 Sys-default 安全组，可省略该步骤。

实例信息

实例名称 kafka- [编辑]

状态 运行中

实例ID [复制]

版本 2.3.0

实例类型 单机

规格 1u2g.single

分区上限 10 个

容量阈值策略 自动删除 生产受限

转储 关闭 [编辑]

Kafka自动创建Topic 关闭

创建时间 2022/02/24 11:31:56 GMT+08:00

描述 -- [编辑]

连接信息

用户名 --

Kafka SASL_SSL 关闭 当前实例暂不支持动态开启/关闭SASL_SSL

内网连接地址 IPv4 [复制] 25.9092

公网访问 已关闭

网络

可用区 可用区2

虚拟私有云 vpc-

子网 subnet-

安全组

Sys-default

sg-69bd

Sys-default

3. 打开公网访问，从可用弹性 IP 中选择公网 IP 进行绑定，并确定。

连接信息

用户名 --

Kafka SASL_SSL 关闭 当前实例暂不支持动态开启/关闭SASL_SSL

内网连接地址 IPv4 [复制] 25.9092

公网访问 已关闭

弹性IP地址

--请选择--

☐ (全选)
 ☐ [弹性IP地址]

创建弹性IP ✓ ✗

网络

4. 等待公网绑定结束后，即可获取到公网访问地址，SASL_SSL 模式的设置可参见 [华为云帮助文档](#)。

注意

华为云 Kafka 实例不支持动态修改 SASL_SSL 模式，请重新创建实例，并在创建实例时开启 SASL_SSL 认证。

连接信息

用户名

--

Kafka SASL_SSL

关闭 当前实例暂不支持动态开启/关闭SASL_SSL

内网连接地址

IPv4 25:9092

公网访问

已开启

公网连接地址

:9094

步骤2：配置腾讯云数据连接器 Kafka 连接器连接属性

1. 在 腾讯云数据连接器控制台 上单击新建应用，选择空白应用进行创建。

深度集成

默认项目

搜索应用、流、组件及配置

新手指引 帮助文档

应用管理

添加应用 导入应用 应用创建教学

应用名称	创建时间	更新时间	运行/调试版本	操作
应用名称	2022-04-12 17:18:56	2022-04-12 17:18:59	-	配置 修改名称 删除 导出
同构信息流连接器_连接器_连接器	2022-04-08 13:34:11	2022-04-08 13:35:27	运行中: 20220408_1.0(广州)	配置 修改名称 删除 导出 运行日志
连接器_连接器	2022-03-30 19:31:29	2022-03-30 19:31:31	-	配置 修改名称 删除 导出

2. 单击左侧的集成流展开选择 NewFlow 在画布中单击 +选择 Apache Kafka 连接器相关操作。

集成流

NewFlow

通用配置

redis

自定义数据类型

连接器版本

自定义脚本

测试用例

Trigger

未配置 单击选择Trigger组件

Apache Kafka

Apache Kafka Producer

Apache Kafka Batch Producer

3. 单击右侧的**新建连接器配置**并按照指引填入相关参数。

新建连接器配置

×

1 选择连接器

>

2 配置连接器

连接器类型 *

Apache Kafka

配置名称 *

test

下一步

取消

4. 单击**下一步**以使用PLAINTEXT接入点为例，在连接器配置参数中，依次填写如下配置：

- 集群地址：填入 Kafka 公网连接地址，可参见 [获取公网访问地址](#)。
- 集群 Kafka 版本：根据购买的 Kafka 实例版本选择对应的版本。
- SASL 安全认证模式：选择 PLAINTEXT。
- 使能 TLS 安全传输协议：选择 false。

编辑连接器配置

×

通用配置

▲

连接配置

集群地址 *

string

:9094

f(x)

Apache Kafka 集群地址，配置的格式:ip:port,ip:port

集群Kafka版本

2.3

f(x)

SASL安全认证模式

PlainText

f(x)

使能TLS安全传输协议

bool

false

f(x)

消费者配置

消费者组 *

string

f(x)

主题 *

string

f(x)

保存

5. 其他 Kafka 参数请按实际情况填写，也可参见 [Apache Kafka 连接器使用指南](#)。

腾讯云 CKafka 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:00

操作场景

腾讯云数据连接器中的 Kafka 连接器支持 PLAINTEXT、SASL_PLAINTEXT、SASL_SCRAM 三种认证方式接入 Kafka 集群，可使用任一被支持的认证方式接入到 Kafka 集群。

前期准备

您需要先开通 CKafka 服务，并设置公网域名访问及对应 ACL 策略。

购买腾讯云 CKafka

详细操作请参见 [创建公网域名接入的 CKafka 实例](#)。

设置 CKafka 接入方式

1. 登录 [CKafka 控制台](#)，在实例列表中选想要设置的实例。

新建 编辑标签 销毁/退还 请输入关键字进行搜索										
☐	ID/名称	监控	状态	可用区	实例类型	配置	网络类型	实例计费模式	公网计费模式	操作
☐	ckafka 容灾 续		健康	广州六区 广州四区	专业版 版本：2.4.1 磁盘类型：SSD云硬盘	Topic数量上限：1000个 Partition数量上限：2000个 峰值带宽：400 MB/s 磁盘容量：4100GB(含 额外购买1100GB)	私有网络	包年包月 2022-04-18到期	包年包月 2022-04-18到期	配置告警 续费 更多
☐	ckafka 容灾 续		健康	广州六区 广州七区	专业版 版本：2.4.1 磁盘类型：SSD云硬盘	Topic数量上限：1000个 Partition数量上限：2000个 峰值带宽：400 MB/s 磁盘容量：4100GB(含 额外购买1100GB)	私有网络	包年包月 2022-04-05到期		配置告警 续费 更多

2. 单击实例的“ID/名称”，进入实例详情页。

ckafka-

基本信息

topic管理

Consumer Group

监控

ACL策略管理

用户管理

基本信息

名称

ID

实例版本

内网IP与端口

地域

可用区

状态

标签

1.1.1

9092

广州

广州三区

健康

接入方式?

添加路由策略

接入类型	接入方式	网络	操作
公网域名接入	SASL_PLAINTEXT	ckafka-	删除

磁盘水位处理?

策略类型	状态	策略描述	当前保留时长/磁盘	操作
动态消息保留策略	☒	当磁盘负载达到 90% 时，向前调整消息保留时间 10%。保底时长 6 小时	查看	配置

3. 单击添加路由策略，在弹出的配置窗口中，路由策略配置为公网域名接入，接入方式选择 SASL_PLAINTEXT。

添加路由策略

×

①

标准版单台 Broker 公网带宽最多支持1Mbps

路由类型

公网域名接入

接入方式

SASL_PLAINTEXT

该接入方式提供用户管理和ACL策略配置，以管理用户访问权限

提交

关闭

版权所有：腾讯云计算（北京）有限责任公司

第288 共732页

4. 单击**提交**，配置完成后，获得公网访问的域名和接口。

添加路由策略

选择SASL_PLAINTEXT接入方式的路由有且仅可以有一条，例如路由类型为公网域名接入选择了SASL_PLAINTEXT接入方式，则创建其他的路由时不可以再选择SASL_PLAINTEXT接入方式。

标准版单台 Broker 公网带宽最多支持1Mbps

路由类型

公网域名接入

接入方式

SASL_PLAINTEXT

该接入方式提供用户管理和ACL策略配置，以管理用户访问权限

提交

关闭

创建 Topic

1. 登录 [CKafka 控制台](#)，在实例列表中选中想要设置的实例，单击实例的“ID/名称”，进入实例详情页。
2. 在实例详情页，单击页面顶部的 **Topic 管理**，单击**新建**。
3. 在编辑 Topic 窗口中，选择分区数和副本数等信息。

新建Topic

名称

test

备注

选填，请输入备注信息

分区数①

12500

-

10

+

个

单个Topic支持最大分区数：2500

副本数①

2个副本

3

选择n个副本时，最多允许有(n-1)台broker宕机
实例支持最大分区*副本数：60，当前额度已用2个，实例还可最多创建29个2副本分区
如需更多分区，可操作实例升配，具体规则见[文档](#)

标签

+ 添加

标签用于从不同维度对资源分类管理。如现有标签不符合您的要求，请前往控制台[管理标签](#)

预设ACL策略

[展示高级配置](#)

提交

关闭

设置 CKafka 登录用户

1. 在 CKafka 实例详情页面中，单击**用户管理**，进入用户管理页面。



2. 单击**新建**，依次输入**用户名**、**密码**、**确认密码**。后续腾讯云数据连接器的连接器配置需填写对应字段。

新增用户

用户名

请输入用户名

只能包含字母、数字、下划线、“-”、“.”

密码

请输入用户密码

只能包含字母、数字、下划线、“-”、“.”

确认密码

请再次输入用户密码

提交

关闭

设置 ACL 策略（可选）

1. 进入对应实例的 ACL 策略管理页面。



2. 单击**批量设置**或单击某一 Topic 右侧的**编辑 ACL 策略**链接，进入新建 ACL 策略。

新增ACL策略

ACL策略示例：允许/拒绝 用户 user 通过 ip 读/写 topic资源

匹配方式

批量勾选

预设规则

选择Topic

请选择

已选择 (1)

ACL策略

操作权限	用户	IP	策略	
允许	kafka	请输入IP, 支持“;”分隔, 默认*	写	
允许	kafka	请输入IP, 支持“;”分隔, 默认*	读	
拒绝	ckafka	请输入IP, 支持“;”分隔, 默认*	写	

添加规则

3. 勾选要设置 ACL 策略的 Topic，并在下方 ACL 策略栏中针对用户和 IP 配置读写策略。

说明

配置 IP 地址为 * 表示允许全部 IP 访问。

接入配置

配置腾讯云数据连接器 Kafka 连接器连接属性步骤如下：

1. 在 [腾讯云数据连接器控制台](#) 上单击**添加应用**，选择空白应用进行创建。

深度集成

默认项目

搜索应用、流、组件及配置

新手指引 帮助文档

应用管理

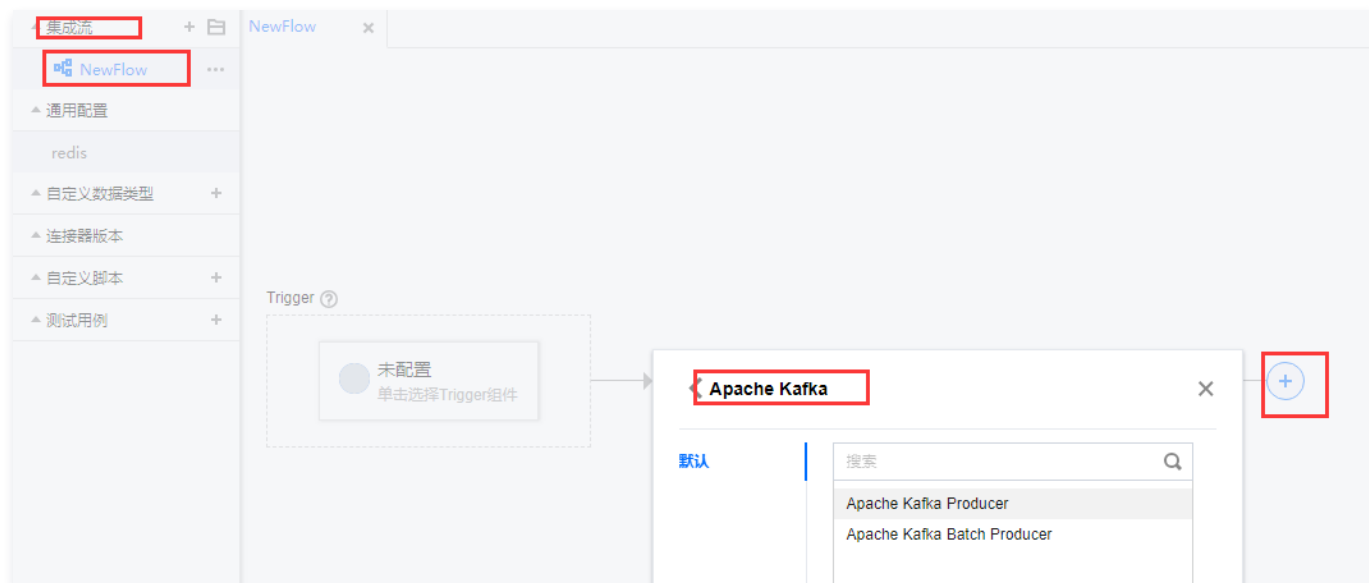
添加应用

导入应用

应用创建教学

应用名称	创建时间	更新时间	运行/调试版本	操作
测试_应用	2022-04-12 17:18:56	2022-04-12 17:18:59	-	配置 修改名称 删除 导出
同密信集成_连接器_1.0.0	2022-04-08 13:34:11	2022-04-08 13:35:27	运行中: 20220408_1.0(广州)	配置 修改名称 删除 导出 运行日志
测试_连接器	2022-03-30 19:31:29	2022-03-30 19:31:31	-	配置 修改名称 删除 导出

2. 单击左侧的集成流展开选择 **NewFlow** 在画布中单击 + 选择 **Apache Kafka** 连接器相关操作。



3. 单击右侧的新建连接器配置并按照指引填入相关参数。



4. 单击下一步以使用 SASL/PLAINTEXT 接入点为例，在连接器配置参数中，依次填写如下配置：

- 集群地址：填入 CKafka 公网接入域名，可参见 [获得公网访问的域名和接口](#)。
- 集群 Kafka 版本：根据购买的 Kafka 实例版本选择对应的版本。
- SASL 安全认证模式：选择 SASL/PLAINTEXT。
- SASL 用户名：CKafka 此处的填写规则为 **CKafka实例ID#CKafka用户名**，假设 CKafka实例ID为ckafka-instance1，[设置 CKafka 登录用户](#) 中设置的用户名为 kafkaUser，则此处用户名应填写：**ckafka-instance1#kafkaUser**。
- SASL 密码：请填写 [设置 CKafka 登录用户](#) 中创建的用户密码。

- 使能 TLS 安全传输协议：选择 false。

编辑连接器配置

通用配置

连接配置

集群地址 *	string ▼ ckafka [redacted] f(x)
Apache Kafka 集群地址，配置的格式:ip:port,ip:port	
集群Kafka版本	1.1 ▼ f(x)
SASL安全认证模式	SASL/PlainText ▼ f(x)
SASL用户名	string ▼ ckafka [redacted] f(x)
SASL密码	 f(x)
使能TLS安全传输协议	bool ▼ false ▼ f(x)

5. 其他 Kafka 参数请根据实际情况填写，也可参见 [Apache Kafka 连接器使用指南](#)。

Database 连接器

Database 连接器使用指南

最近更新时间：2024-10-10 15:55:21

简介

Database 连接器可连接第三方关系型数据库系统并执行 SQL 操作。用户通过连接器配置来配置数据库的连接参数，配置成功后便可执行对应的数据库操作。Database 连接器目前支持的数据库有：MySQL、Oracle、PostgreSQL、SQL Server。

连接器配置

Database 连接器配置参数

参数	数据类型	描述	是否必填	默认值
数据库类型	enum	数据库类型，支持 MySQL、Oracle、PostgreSQL、SQL Server	是	-
地址	string	数据库地址	是	-
端口号	int	数据库端口号	是	-
用户名	string	用于连接数据库的用户名	是	-
密码	string	用于连接数据库的用户密码	是	-
超时时间	int	数据库连接超时时间，时间单位：秒，最长超时时间为30秒	否	5

连接器配置界面如下：

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

通用

数据库类型

请选择

地址

127.0.0.1

数据库地址

端口号

int

 请输入整数

端口号范围：0-65535

用户名

root

账户名称

数据库

ipaas

数据库库名

密码

123456

账户密码

超时时间(秒)

int

 5

最长超时时间：30秒

高级配置

安全网关

无

操作说明

Database 连接器目前支持查询、插入、更新、删除、批量插入、批量合并、批量删除、存储过程（目前仅支持 SQL Server）操作。在配置连接器和集成流数据流转过程中，数据库字段类型与数据连接器 iPaaS 字段类型的映射关系见 [附录](#)。

查询操作

SQL模式				
参数配置				
查询逻辑配置				
参数	数据类型	描述	是否必填	默认值
操作模式	enum	操作模式，支持简单模式-单表查询、简单模式-多表查询和 SQL 模式	是	SQL 模式
查询语句	string	SQL 模式参数，SQL 语句，支持两种写法： 1. 原生 SQL。例如："select * from p where age=12" 2. 嵌入占位符的 SQL，使用参数化输入（冒号+参数），可防止 SQL 注入。例如："select * from p where age = :age"，在“输入参数”中，填入变量名 key 为"age"，value 值为"12"，此时待执行的查询语句为“select * from p where age=12”	是	-

输入参数	dict	SQL 模式参数，输入参数列表，列表元素为字典，key 对应“查询语句”中的参数化变量名，key 值需要与参数化变量名一致，value 为该变量的值	否	-
------	------	--	---	---

配置

调试

备注

查询逻辑配置

操作模式

SQL模式

支持简单模式-单表查询、简单模式-多表查询和SQL模式

查询语句

string

select * from p where age=:age

f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数

☐ 表达式

value

操作

+添加变量

请填入SQL语句中的参数，key为参数名称，value为参数值

输出

查询操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，键表示数据库字段名称，值表示数据库字段值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  {
    "book_auth": "aaa",
    "book_id": 1,
    "book_name": "a",
    "book_price": "25.16"
  },
  {
    "book_auth": "bbb",
    "book_id": 2,
    "book_name": "b",
    "book_price": "15.26"
  }
]
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Login error: mssql: Cannot open database \"test\" that was requested by the login.
Using the user default database \"master\" instead."
}
```

案例

1. 添加 Database 连接器组件，选择查询操作。



2. 新建连接器配置，填写配置参数，单击**测试连接**，测试连接器配置是否正确。

编辑连接器配置

✓ 链接配置正确

✓ 选择连接器

>

2 配置连接器

通用

数据库类型 *

Sql Server

地址 *

数据库地址

端口号 *

int

端口号范围：0-65535

用户名 *

账户名称

数据库 *

数据库库名

密码 *

账户密码

超时时间(秒)

int

5

最长超时时间：30秒

高级配置

安全网关

无

测试连接

上一步

保存

3. 在通用配置中，填入 SQL 语句及输入参数。例如：“查询语句”为 `select * from books where book_price > :book_price`，输入参数中 key 为 book_price，value 为30，正确的查询结果如下：

配置
调试
备注

查询逻辑配置

操作模式
SQL模式

支持简单模式-单表查询、简单模式-多表查询和SQL模式

查询语句
string
select * from books where book_pi
f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数
☐ 表达式

value	操作
item_1 ☐ 表达式	删除

+添加变量

请填入SQL语句中的参数，key为参数名称，value为参数值

4. 查询结果：

- 查询成功后，message payload 中包含了查询结果：

输入消息
payload

```
[
  {
    "book_auth": "aaa",
    "book_id": 1,
    "book_name": "a",
    "book_price": "25.16"
  },
  {
    "book_auth": "bbb",
    "book_id": 2,
    "book_name": "b",
    "book_price": "15.26"
  },
  {
    "book_auth": "ccc",
    "book_id": 3,
    "book_name": "c",
    "book_price": "32"
  }
]
```

○ 若查询过程中出现错误，message error 中会包含错误信息：

输出消息 error ▼

```
{
  "Code": "CORE:RUNTIME",
  "Description": "invalid params:book_price"
}
```

简单模式-单表查询

参数配置

查询逻辑配置

参数	数据类型	描述	是否必填	默认值
操作模式	enum	操作模式，支持简单模式-单表查询、简单模式-多表查询和 SQL 模式	是	SQL 模式
表选择	enum	简单模式-单表查询参数，待查询的数据表	是	-
字段选择	list	简单模式-单表查询参数，查询的数据表字段	是	-
过滤条件	list	简单模式-单表查询参数，查询的过滤条件	否	-

输出配置

参数	数据类型	描述	是否必填	默认值
输出模式	enum	查询所得数据的输出模式，支持普通模式和 RecordSet 模式	是	普通模式
是否缓存	bool	输出模式为 RecordSet 模式时的参数，输出的 RecordSet 数据是否支持缓存	否	false
分区数量	int	输出模式为 RecordSet 模式时的参数，输出的 RecordSet 数据分区数设置，范围[1,10]	是	1

输出

查询操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，键表示数据库字段名称，值表示数据库字段值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Database 连接器组件，选择查询操作。
- 2. 新建连接器配置或选择已创建的连接器配置。

3. 配置中的“表选择”选择数据表“books”，“字段选择”填写“book_id”和“book_name”字段，输出模式选择普通输出，过滤条件中填写如下内容：

配置

调试

备注

查询逻辑配置

操作模式

简单模式-单表查询

支持简单模式-单表查询、简单模式-多表查询和SQL模式

表选择

books

字段选择

字段名	字段类型	操作
book_id	int	删除
book_name	string	删除

+新增字段

过滤条件

序号	字段名	操作	目标值	
1	book_id	>	int	3740

+添加变量

输出配置

输出模式

普通模式

输出模式支持RecordSet模式和普通模式

4. 查询成功后，message payload 中包含了查询结果：

输出消息 payload

```
[
  {
    "book_id": 3740,
    "book_name": "ggg"
  }
]
```

简单模式-多表查询

参数配置

通用：

参数	数据类型	描述	是否必填	默认值
----	------	----	------	-----

操作模式	enum	操作模式，支持简单模式-单表查询、简单模式-多表查询和 SQL 模式	是	SQL 模式
------	------	------------------------------------	---	--------

查询逻辑配置：

参数	数据类型	描述	是否必填	默认值
连接类型	enum	简单模式-多表查询参数，支持 INNER JOIN、LEFT JOIN、RIGHT JOIN	是	
连接条件	list	简单模式-多表查询参数，数据表间 join 查询的连接条件	是	-
过滤条件	list	简单模式-单表查询参数，查询的过滤条件	否	-
字段选择	list	简单模式-多表查询参数，查询的数据表字段	是	-

输出配置：

参数	数据类型	描述	是否必填	默认值
输出模式	enum	查询所得数据的输出模式，支持普通模式和 RecordSet 模式	是	普通模式
是否缓存	bool	输出模式为 RecordSet 模式时的参数，输出的 RecordSet 数据是否支持缓存	否	false
分区数量	int	输出模式为 RecordSet 模式时的参数，输出的 RecordSet 数据分区数设置，范围[1,10]	是	1

输出

查询操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，键表示数据库字段名称，值表示数据库字段值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Database 连接器组件，选择查询操作。
2. 新建连接器配置或选择已创建的连接器配置。

3. 在配置中填写如下配置：

查询逻辑配置

操作模式

简单模式-多表查询

支持简单模式-单表查询、简单模式-多表查询和SQL模式

连接类型

INNER JOIN

支持INNER JOIN、LEFT JOIN、RIGHT JOIN

连接条件

关联数据表	关联数据表字段	待关联数据表	待关联数据表...	操作
student	class_name	class	class_name	删除

+添加条件

过滤条件

数据表	数据表字段	操作符	输入值
student	student_id	>	int 2

+添加条件

字段选择

数据表	数据表字段	字段类型	操作
student	student_id	int	删除
class	class_teacher	string	删除

+新增字段

输出配置

输出模式

普通模式

查询成功后，message payload 中包含了查询结果：

输出消息 payload

```
[
  {
    "class.class_teacher": "高阳",
    "student.student_id": 3
  }
]
```

Recordset 输出模式

输出

查询操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message属性	值
payload	执行成功后，payload 为 RecordSet 类型数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Database 连接器组件，选择查询操作。
2. 新建连接器配置或选择已创建的连接器配置。
3. 在配置中填写如下配置，输出模式选择 **RecordSet 模式**：

查询逻辑配置

操作模式

简单模式-单表查询

支持简单模式-单表查询、简单模式-多表查询和SQL模式

表选择

student

字段选择

字段名	字段类型	操作
student_id	int	删除
student_age	string	删除

+新增字段

过滤条件

序号	字段名	操作	目标值
1	student_id	>	int 1

+添加变量

查询成功后，message payload 中包含了查询结果，该查询结果为 RecordSet 类型：

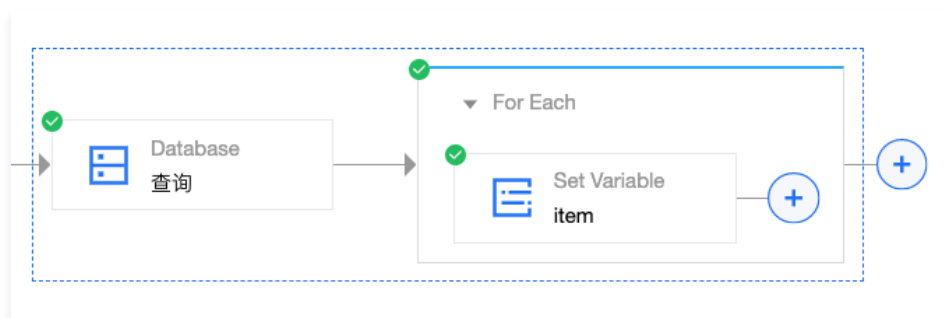
输出消息 payload ▾

```

{
  "id": 37,
  "partitions": 1,
  "records": [
    [
      2,
      "14"
    ],
    [
      3,
      "15"
    ],
    [
      4,
      "15"
    ]
  ],
  "schema": [
    "student_id(int)",
    "student_age(string)"
  ]
}

```

可以通过“For Each”组件对 RecordSet 数据进行迭代：



迭代结果如下：

输出消息 payload ▾ message0 ▾

message0

```

{
  "data": [
    2,
    "14"
  ],
  "schema": [
    "student_id(int)",
    "student_age(string)"
  ]
}

```

message1

message2

下载

插入操作

SQL 模式

参数配置

参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式
插入语句	string	SQL 模式参数，SQL 语句，支持两种写法： - 原生 SQL - 嵌入占位符的 SQL，使用参数化输入（冒号+参数），可防止 SQL 注入	是	-
输入参数	dict	SQL 模式，输入参数列表，列表元素为字典，key 对应“插入语句”中的参数化变量名，key 值需要与参数化变量名一致，value 为该变量的值	否	-

配置

调试

备注

插入逻辑配置

操作模式

SQL模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

插入语句

string

insert into p (age) values (:age)

f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数

☐ 表达式

value

操作

+添加变量

请填入SQL语句中的参数，key为参数名称，value为参数值

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含 “rowsAffected” 字段，当为 MySQL 数据库时，会额外包含 “lastId” 字段；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：MySQL 数据库的插入操作，执行成功后，message payload 值如下：

```
{
  "lastId": 1,
```



```
{
  "rowsAffected": 1
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Login error: mssql: Cannot open database \"test\" that was requested by the login.
Using the user default database \"master\" instead."
}
```

案例

1. 添加 Database 连接器组件，选择插入操作。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入 SQL 语句及输入参数。例如：“插入语句”为

`insert into books (book_auth, book_name, book_price) values (:book_auth, :book_name, :book_price)`，输入参数使用 Dataway 格式编写。

```
def dw_process(msg):
    return {
        'book_auth': 'test',
        'book_name': 'ipaas',
        'book_price': 100,
    }
```

4. 插入结果：

- 插入成功后，message payload 中包含了执行结果：

输出消息 payload

```
{
  "rowsAffected": 1
}
```

- 若插入过程中出现错误，message error 中会包含错误信息：

输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "mssql: Invalid column name 'book_aut'."
}
```

简单模式

参数配置

通用：

参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式

插入逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	enum	简单模式参数，待操作的数据表	是	-
字段配置	list	简单模式参数，待插入的字段信息，包括字段名称和字段值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含 “rowsAffected” 字段，当为 MySQL 数据库时，会额外包含 “lastId” 字段；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Database 连接器组件，选择插入操作。
- 2. 新建连接器配置或选择已创建的连接器配置。
- 3. 在配置中填写如下配置：

插入逻辑配置

操作模式

简单模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

表选择

books

字段配置

数据表字段	输入值	操作
book_name	string eee	f(x) 删除
book_price	float 6.66	f(x) 删除
book_auth	string e	f(x) 删除

+新增字段

4. 插入成功后，message payload 如下：

配置

调试

备注

查询逻辑配置

操作模式

简单模式-单表查询

支持简单模式-单表查询、简单模式-多表查询和SQL模式

表选择

books

字段选择

字段名	字段类型	操作
book_id	int	删除
book_name	string	删除

+新增字段

过滤条件

序号	字段名	操作	目标值	
1	book_id	>	int	3740

+添加变量

输出配置

输出模式

普通模式

输出模式支持RecordSet模式和普通模式

更新操作

SQL模式				
参数配置				
参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式
更新语句	string	SQL 模式参数，SQL 语句，支持两种写法： - 原生 SQL - 嵌入占位符的 SQL，使用参数化输入（冒号+参数），可防止 SQL 注入	是	-
输入参数	dict	SQL 模式，输入参数列表，列表元素为字典，key 对应“更新语句”中的参数化变量名，key 值需要与参数化变量名一致，value 为该变量的值	否	-

更新逻辑配置

操作模式

SQL 模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

更新语句

string

update p set n=1 where age=:age

f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数

☐ 表达式

value	操作
+添加变量	

请填入SQL语句中的参数，key为参数名称，value为参数值

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含 “rowsAffected” 字段；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如，执行成功后，message payload 值如下：

```
{
  "rowsAffected": 1
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Login error: mssql: Cannot open database \"test\" that was requested by the login. Using the user default database \"master\" instead."
}
```

案例

1. 添加 Database 连接器组件，选择更新操作。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入 SQL 语句及输入参数。例如，“更新语句”为
update books set book_price = :book_price where book_name = :book_name ，输入参数如下：

```
def dw_process(msg):
    return {
        'book_price': 200,
```

```
'book_name': 'ipaas'
}
```

4. 更新结果：

- 更新成功后，message payload 中包含了执行结果：

```
输出消息  payload
{
  "rowsAffected": 1
}
```

- 若更新过程中出现错误，message error 中会包含错误信息：

```
输出消息  error
{
  "Code": "CORE:RUNTIME",
  "Description": "mssql: Invalid column name
'book_nme'."
}
```

简单模式

参数配置

通用：

参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式

更新逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	enum	简单模式参数，待操作的数据表	是	—
字段配置	list	简单模式参数，待更新的字段名称和字段值	是	—
过滤条件	list	简单模式参数，更新条件	否	—

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含 “rowsAffected” 字段；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Database 连接器组件，选择更新操作。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填入如下配置：

更新逻辑配置

操作模式

简单模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

表选择

books

字段配置

数据表字段	输入值	操作
book_name	string xiaohua f(x)	删除
book_price	string 8.88 f(x)	删除

+新增字段

过滤条件

序号	字段名	操作	目标值
1	book_name	==	string xiaoming f(x)

+添加变量

- 4. 更新成功后，message payload 中包含了执行结果：

更新逻辑配置

操作模式

SQL模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

更新语句

string update p set n=1 where age=:age f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数

☐ 表达式

value	操作
-------	----

+添加变量

请填入SQL语句中的参数，key为参数名称，value为参数值

删除操作

SQL模式

参数配置

参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式
删除语句	string	SQL 模式，SQL 语句，支持两种写法： - 原生 SQL - 嵌入占位符的 SQL，使用参数化输入（冒号+参数），可防止 SQL 注入	是	-
输入参数	dict	SQL 模式，输入参数列表，列表元素为字典，key 对应“删除语句”中的参数化变量名，key 值需要与参数化变量名一致，value 为该变量的值	否	-

删除逻辑配置

操作模式

SQL模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写结构化SQL语句

删除语句

string

delete from p where age=:age

f(x)

同时支持原生SQL及参数化输入。使用参数化(冒号+参数名)输入，可防止SQL注入

输入参数

☐ 表达式

value

操作

+添加变量

请填入SQL语句中的参数，key为参数名称，value为参数值

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含“rowsAffected”字段；执行失败后，payload为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "rowsAffected": 1
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "..."
}
```

```
"Description": "Login error: mssql: Cannot open database \"test\" that was requested by the login.
Using the user default database \"master\" instead."
}
```

案例

1. 添加 Database 连接器组件，选择删除操作。
2. 新建连接器或选择已创建的连接。
3. 在通用配置中，填入 SQL 语句及输入参数。例如：“删除语句”为 `delete from books where book_name = :book_name`，输入参数如下：

```
def dw_process(msg):
    return {
        'book_name': 'ipaas'
    }
```

4. 删除结果：

- 删除成功后，message payload 中包含了执行结果：

输出消息 payload ▼

```
{
  "rowsAffected": 1
}
```

- 若执行过程中出现错误，message error 中会包含错误信息：

输出消息 error ▼

```
{
  "Code": "CORE:RUNTIME",
  "Description": "mssql: Invalid object name 'book'."
}
```

简单模式

参数配置

通用：

参数	数据类型	描述	是否必填	默认值
操作模式	enum	支持简单模式和 SQL 模式；简单模式可辅助编写操作语句，SQL 模式下可编写结构化 SQL 语句	是	SQL 模式

删除逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	enum	简单模式参数，待操作的数据表	是	—
过滤条件	list	简单模式参数，删除条件	否	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，包含 “rowsAffected” 字段；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Database 连接器组件，选择删除操作。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入如下配置：

删除逻辑配置

操作模式

简单模式

支持简单模式和SQL模式；简单模式可辅助编写操作语句，SQL模式下可编写SQL语句

表选择

books

过滤条件

序号	字段名	操作	目标值
1	book_id	==	int1f(x)

+添加变量

4. 删除成功后，message payload 中包含了执行结果：

输出消息 payload

```
{  "rowsAffected": 1}
```

批量插入操作

参数配置

输入配置：

参数	数据类型	描述	是否必填	默认值
输入数据集	Recordset 类型	输入数据集，若未填写，默认为 message 的 payload	是	—
输入数据集 Schema 校准	List	对输入数据集进行字段校准	否	—

插入逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	Enum	待操作的数据表	否	—
过滤条件	List	批量更新的过滤条件	是	—

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为前一个组件的 payload；执行失败后，payload为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Database 连接器组件，选择批量插入操作。
2. 新建连接器或选择已创建的连接器。
3. 通过 RecordSet Encoder 生成 RecordSet 类型的数据，在 Encoder 组件中设置如下 Schema：

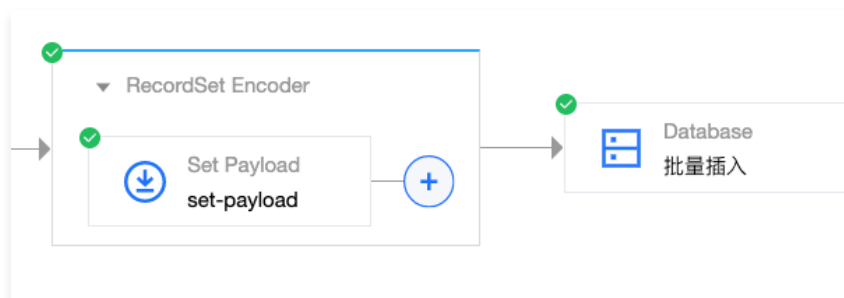
序号	字段类型	字段名	显示名称	操作
1	string	name	name	删除
2	float	price	price	删除
3	string	auth	auth	删除

+添加变量

4. 在 Encoder 组件选择 Set Payload 组件，按如下配置设置：

```
def dw_process(msg):
    return [['xiaomi',66.66,'e'], ['xiaoming',66.66,'f'], ['xiaohua',66.66,'g']]
```

5. 测试组件如下：



6. 构造好 RecordSet 数据后，输入数据集如下：

```
def dw_process(msg):
    return msg.payload
```

7. Database组件通过数据表字段和 RecordSet 数据字段的映射关系来构造插入的数据信息，配置如下：

配置

备注

输入配置

输入数据集

Record

def dw_process(msg): return ms;

输入数据集Schema校准

字段名	字段类型	操作
+新增字段		

插入逻辑配置

表选择

books

字段映射

表字段	输入字段	操作
book_name	name	删除
book_price	price	删除
book_auth	auth	删除
+新增字段		

忽略中间错误

bool

true

f(x)

8. 批量插入成功后，message 的 error 信息为空。

批量合并操作

参数配置

输入配置：

参数	数据类型	描述	是否必填	默认值
输入数据集	Recordset 类型	输入数据集，若未填写，默认为 message 的 payload	是	-
输入数据集 Schema 校准	list	对输入数据集进行字段校准	否	-

合并逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	enum	待操作的数据表	否	-
过滤条件	list	批量更新的过滤条件	是	-
字段映射	enum	字段映射，数据表字段和 Recordset 类型数据字段的映射，插入	是	-

		数据表字段的值为 Recordset 类型数据字段的值		
只执行插入	bool	数据不存在时执行插入操作	是	false
只执行更新	bool	数据存在时执行更新操作	是	false

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为前一个组件的 payload；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 新建连接器或选择已创建的连接器。
2. 通过 RecordSet Encoder 组件生成 RecordSet 类型的数据。
3. 构造好 RecordSet 数据后，添加 Database 组件的“批量合并”操作，数据表字段和RecordSet数据字段的关系构成了合并的过滤条件，数据表字段和RecordSet 数据字段的映射构成了合并的字段映射；Database 组件配置如下：

配置

备注

输入配置

输入数据集

输入数据集

Record

def dw_process(msg):
return ms

输入数据集Schema校准

字段名	字段类型	操作
+新增字段		

合并逻辑配置

表选择

books

过滤条件

表字段	操作符	输入字段	操作
book_name	==	name	删除
+添加条件			

字段映射

表字段	输入字段	是否更新	操作
book_price	price	是	删除
+新增字段			

只执行插入

bool

false

f(x)

只执行更新

bool

false

f(x)

忽略中间错误

bool

true

f(x)

4. 批量插入成功后，message 的 error 信息为空。

批量删除操作

参数配置

输入配置：

参数	数据类型	描述	是否必填	默认值
输入数据集	Recordset 类型	输入数据集，若未填写，默认为 message 的 payload	是	-
输入数据集 Schema 校准	List	对输入数据集进行字段校准	否	-

删除逻辑配置：

参数	数据类型	描述	是否必填	默认值
表选择	Enum	待执行操作的数据表	否	-
过滤条件	List	批量删除的条件	是	-

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为前一个组件的 payload；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 新建连接器或选择已创建的连接器。
- 2. 通过 “RecordSet Encoder” 组件生成 RecordSet 类型的数据。
- 3. 构造好RecordSet数据后，添加 Database 组件的 “批量删除” 操作，数据表字段和 RecordSe t数据字段的关系构成了合并的过滤条件，Database 组件配置如下：

输入配置

输入数据集

输入数据集

Record▼

请输入表达式

🗑

输入数据集Schema校准

字段名	字段类型	操作
+新增字段		

删除逻辑配置

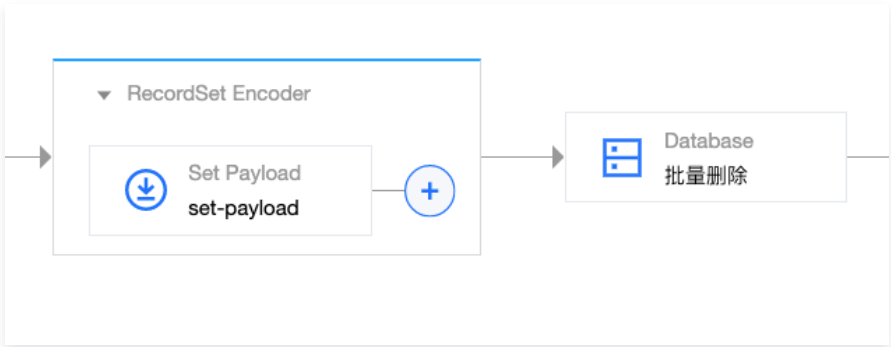
表选择

books▼

过滤条件

表字段	操作符	输入字段	操作
book_name▼	==▼	name▼	删除
+添加条件			

4. 组件如下：



5. 批量插入成功后，message 的 error 信息为空。

存储过程操作

参数配置

参数	数据类型	描述	是否必填	默认值
存储过程名称	string	待调用存储过程的名称		-
存储过程输入参数	string	待调用存储过程的输入参数，输入参数顺序必须和存储过程传入参数顺序一致，使用参数化(冒号+参数名)输入，参数间以英文逗号相隔	是	-
输入参数	dict	输入参数列表，列表元素为字典，key对应“存储过程输入参数”中的变量名，key值需要与参数化变量名一致，value 为该变量的值	否	-
存储过程输出参数	list	待调用存储过程的输出参数信息，输出参数顺序需要和存储过程的输出参数一致，列表添加时需要输入“字段名称”及“字段类型”信息；若使用表达式输入，列表元素字段名称命名为'fieldName'，字段类型为'fieldType'		-

配置

调试

备注

通用

存储过程名称

请输入字符串

存储过程输入参数

string

:param1,:param2,:param3

f(x)

输入参数顺序必须和存储过程传入参数顺序一致，使用参数化(冒号+参数名)输入，参数间以英文逗号相隔

输入参数

☒ 列表添加

☐ f(x)

value

操作

+添加

请填入存储过程输入参数中的参数，key为参数名称，value为参数值

存储过程输出参数

☒ 列表添加

☐ f(x)

value

操作

+添加

请填入存储过程输出参数信息，输出参数顺序需要和存储过程的输出参数一致；若使用表达式输入，列表元素字段名称命名为'fieldName'，字段类型为'fieldType'

输出

操作执行成功后，输出结果会保存在message消息体的payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，存储过程输出的payload取决于输出参数，payload 为 list 类型，list 成员为 dict 类型，键表示输出参数字段名称，值表示输出参数字段值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下，其中 “id” 和 “book_name” 为指定的输出参数字段：

```
[
  {
    "id": 1,
    "book_name": "a",
  },
  {
    "id": 2,
    "book_name": "b",
  }
]
```



```
]
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Login error: mssql: Cannot open database \"test\" that was requested by the login. Using
the user default database \"master\" instead."
}
```

案例

1. 添加 Database 连接器组件，选择存储过程操作。



2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入参数信息。例如：“存储过程”为 getBookId，输入参数 dataway 表达式如下：

```
def dw_process(msg):
    return {
        'name': 'aaa',
        'price': 0,
    }
```

- 输出参数表达式如下：

```
def dw_process(msg):
    return [
        {'fieldName': 'id', 'fieldType': 'INT'},
        {'fieldName': 'book_name', 'fieldType': 'VARCHAR'}
    ]
```

- 界面如图：

配置

调试

备注

通用

存储过程名称

getBookId

存储过程输入参数

string

:name,:price

f(x)

输入参数顺序必须和存储过程传入参数顺序一致，使用参数化(冒号+参数名)输入，参数间以英文逗号相隔

输入参数

☐ 列表添加

☒ f(x)

def dw_process(msg):

return {

请填写入存储过程输入参数中的参数，key为参数名称，value为参数值

存储过程输出参数

☐ 列表添加

☒ f(x)

def dw_process(msg):

return [

请填写入存储过程输出参数信息，输出参数顺序需要和存储过程的输出参数一致；若使用表达式输入，列表元素字段名称命名为'fieldName'，字段类型为'fieldType'

4. 执行结果：

- 执行成功后，message payload 中包含了执行结果：

输出消息 payload

```
[
  {
    "book_name": "a",
    "id": 1
  }
]
```

- 若执行过程中出现错误，message error 中会包含错误信息：

输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "mssql: Error converting data type nvarchar to float."
}
```

附录

Database 连接器会根据数据库字段类型将数据库字段值转换为数据连接器的 **核心类型**：

MySQL 数据库与数据连接器的字段映射：

MySQL 字段	iPaaS 数据类型
TINYINT	整数

SMALLINT	整数
MEDIUMINT	整数
INT	整数
BIGINT	整数
DECIMAL	十进制
NUMERIC	十进制
FLOAT	浮点数
DOUBLE	浮点数
BIT	整数
DATE	日期
TIME	时间
DATETIME	时刻
TIMESTAMP	时刻
YEAR	整数
CHAR	字符串
VARCHAR	字符串
BINARY	字节数组
VARBINARY	字节数组
TINYBLOB	字节数组
MEDIUMBLOB	字节数组
BLOB	字节数组
LOB	字节数组
TINYTEXT	字符串
MEDIUMTEXT	字符串
TEXT	字符串
LONGTEXT	字符串
JSON	字符串

Database 连接器不支持 MySQL 的空间数据类型、ENUM 类型、SET 类型的处理，对于不支持的 MySQL 字段类型统一按照字符串类型进行映射。

Oracle 数据库与数据连接器的字段映射：

Oracle 字段	iPaaS 数据类型
CHAR	字符串
NCHAR	字符串
VARCHAR	字符串
VARCHAR2	字符串
NVARCHAR2	字符串
LONG	字符串

NUMBER	十进制
DECIMAL	十进制
FLOAT	浮点数
BINARY_FLOAT	浮点数
BINARY_DOUBLE	浮点数
DATE	日期
TIMESTAMP	时刻
TIMESTAMP WITH TIME ZONE	时刻
TIMESTAMP WITH LOCAL TIME ZONE	时刻
BLOB	字节数组
CLOB	字节数组
NCLOB	字节数组
BFILE	字节数组
RAW	字节数组
LONG RAW	字节数组

Database 连接器不支持 Oracle 的 INTERVAL DAY TO SECOND 类型、INTERVAL YEAR TO MONTH 类型、ROWID 类型、UROWID 类型、ANSI、DB2 和 SQL/DS 数据类型、用户自定义等数据类型的处理，对于不支持的 Oracle 字段类型统一按照字符串类型进行映射。

华为云 MySQL 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:00

操作场景

Database 连接器支持 MySQL 数据库的连接及增删改查等数据库常用操作。

前期准备

您需要开通 MySQL，并已配置好安全组规则和公网 IP。

购买 MySQL实例

购买 [云数据库 RDS for MySQL](#) 服务，根据需求购买云数据库。

❗ 说明

默认内网端口为3306，可自定义，后面安全组的端口配置需根据此处选择的端口号填写。

设置数据库密码

设置密码

现在设置

创建后设置

管理员帐户名

root

管理员密码

.....

...

请妥善保管密码，系统无法获取您设置的密码内容。

确认密码

.....

设置安全组

1. 登录华为云控制台，进入 [安全组](#) 设置。
2. 修改新建安全组或者已有安全组。

安全组 ⓘ

📖 使用指南 [创建安全组](#)

❗ 诚邀您参加虚拟私有云使用体验调研，您宝贵的意见和建议是我们持续提升产品体验的源动力，感谢您的参与！

❗ 安全组是一个逻辑上的分组，为同一个VPC内具有相同安全保护需求并相互信任的云服务器提供访问策略。安全组创建后，用户可以在安全组中定义各种访问规则，当云服务器加入该安全组后，即受到这些访问规则保护。安全组默认出方向放行，并且安全组内的云服务器可以相互访问。

列表中“Sys-default”是系统默认创建的一个安全组，默认安全组的规则是在出方向上的数据报文全部放行，入方向访问受限，安全组内的云服务器无需添加规则即可互相访问。默认安全组您可以直接使用，详情请参见[默认安全组和规则](#)。如果默认安全组不能满足需求，可创建新的安全组，请参见[创建安全组](#)。

名称

安全组规则

关联实例

描述

操作

default

11

1

Default security group

[配置规则](#) | [管理实例](#) | [克隆](#)

新建安全组

1. 单击控制台右上角的创建安全组，进入安全组创建页面，模板选择自定义。

创建安全组

名称

sg-69bd

模板

自定义

描述

入方向不放通任何端口，您可在安全组创建后，根据实际访问需求添加或修改安全组规则。

0/255

查看模板规则

确定

取消

2. 单击配置规则。

提示

安全组创建成功，请添加安全组规则以便能正常访问该安全组关联的实例。

配置规则

暂不配置

3. 添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	3306	0.0.0.0/0	通过公网访问MySQL

注意

若需配置指定的腾讯云数据连接器出口IP访问，请 [提交工单](#) 联系我们。

添加入方向规则 教我设置

安全组入方向规则为白名单（允许），放通入方向网络流量。

安全组

如您要添加多条规则，建议单击[导入规则](#)以进行批量导入。

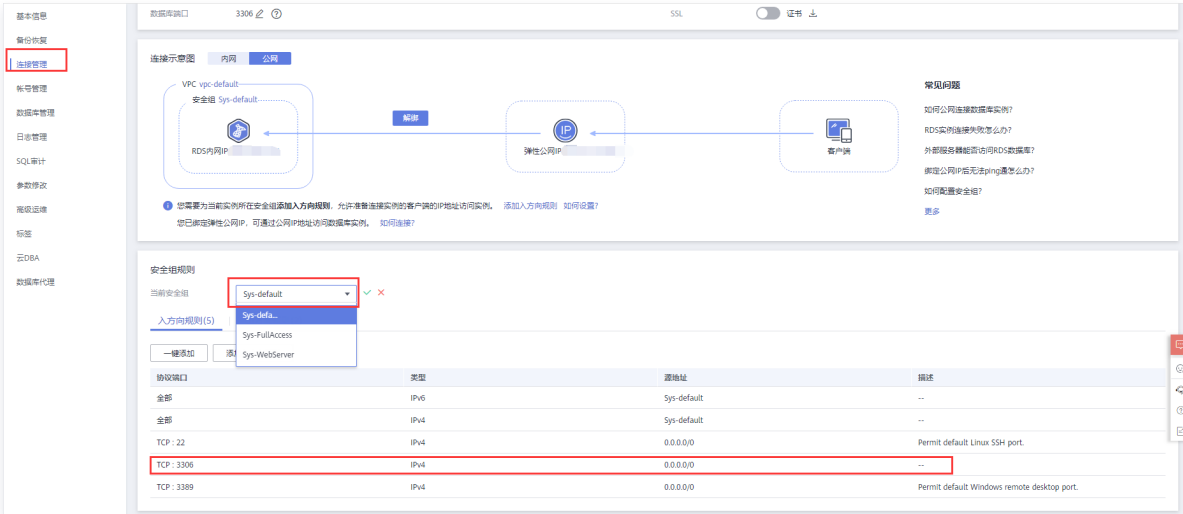
优先级	策略	协议端口	类型	源地址	描述	操作
1	允许	TCP 3306	IPv4	IP地址 0.0.0.0/0		复制 删除

增加1条规则

确定

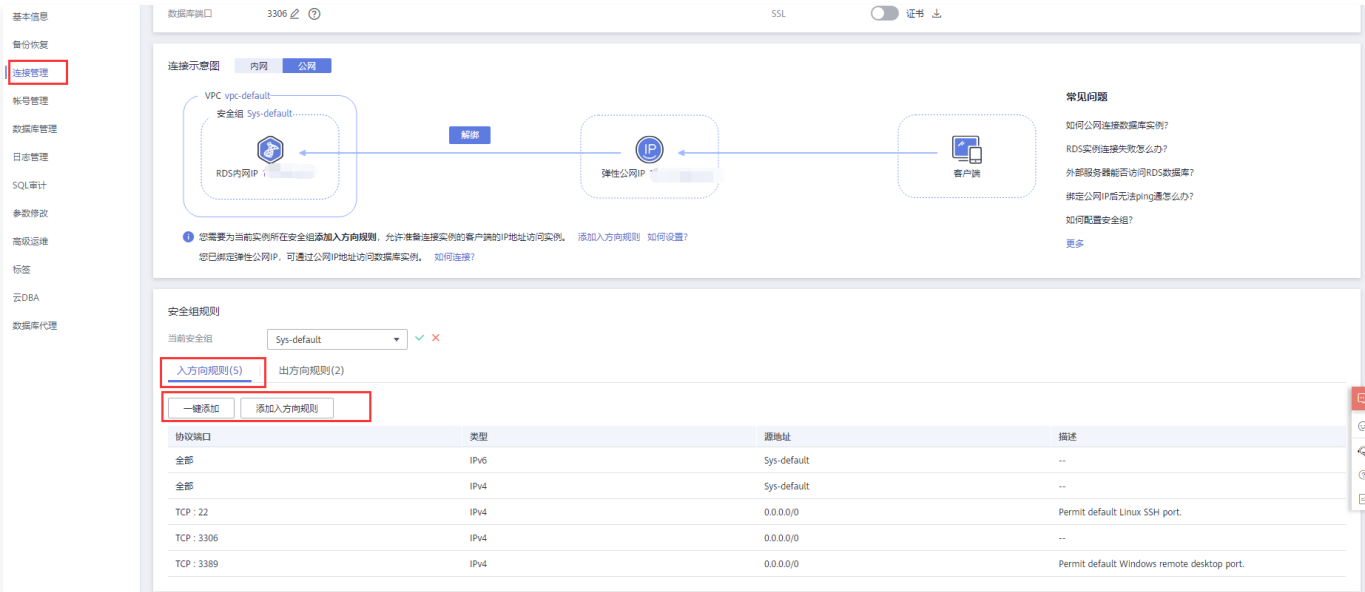
取消

4. 在 **实例管理** 页面，选择目标实例，单击实例名称，进入实例的基本信息页面。在左侧导航栏，单击**连接管理**，可切换为新建的安全组绑定。



修改已有安全组

1. 在 **实例管理** 页面，选择目标实例，单击实例名称，进入实例的基本信息页面。在左侧导航栏，单击**连接管理**，可对默认绑定的安全组规则进行修改。



2. 修改入方向规则，添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	3306	0.0.0.0/0	通过公网访问MySQL

 **注意**

若需配置指定的 iPaaS 出口 IP 访问，请 [提交工单](#) 联系我们。

绑定弹性公网 IP

您可以购买弹性公网 IP 绑定到 MySQL，或者绑定已有的弹性公网 IP 到 MySQL。

购买弹性公网 IP 绑定到 MySQL

1. 单击查看弹性公网 IP，获取弹性公网 IP，单击右上角的购买弹性公网 IP。

注意

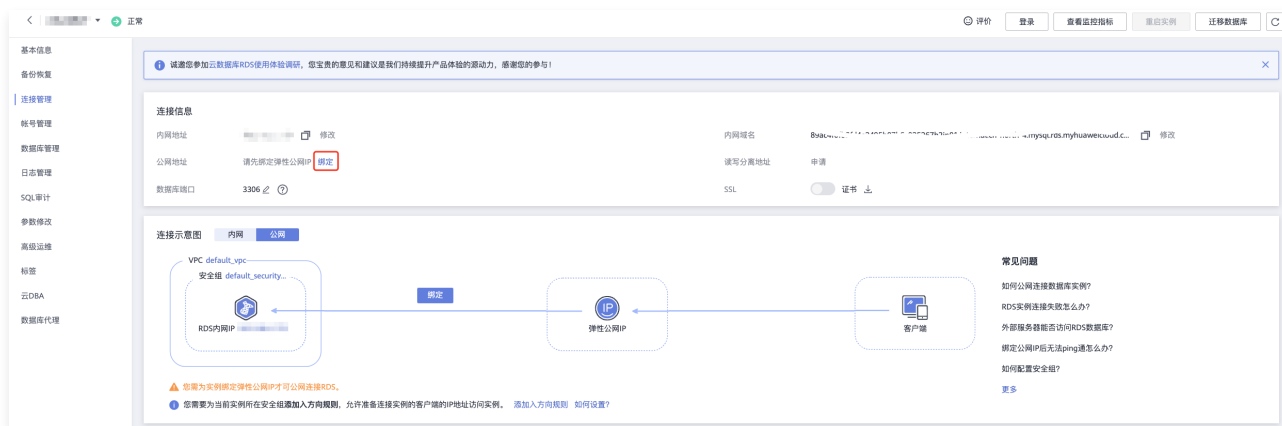
购买弹性公网 IP 需要和 MySQL 处于同一可用区。



2. 按实际情况选择计费模式，设置区域，线路和带宽等参数，单击立即购买。



3. 购买弹性公网 IP 后，在 [实例管理](#) 页面，选择目标实例，单击实例名称，进入实例的基本信息页面。在左侧导航栏，单击连接管理，在连接信息>公网地址处，单击绑定。

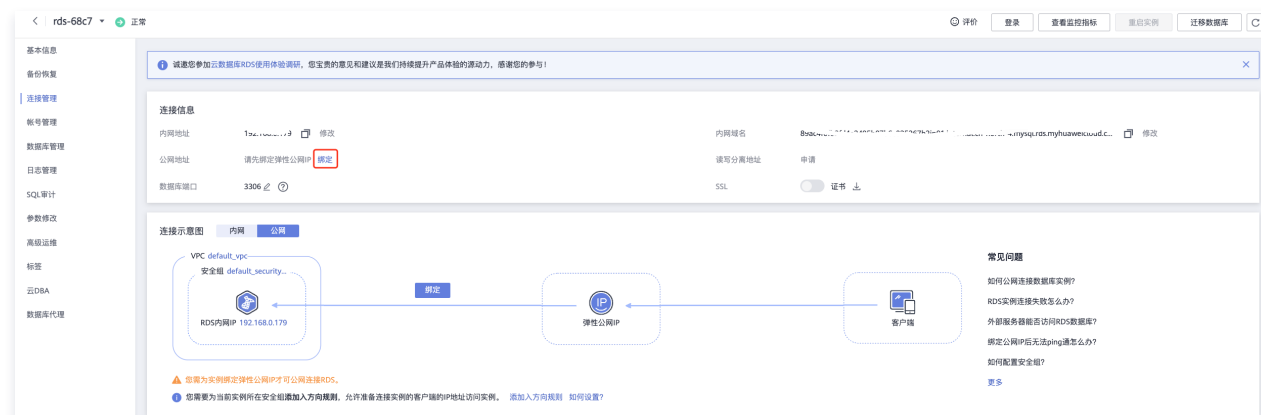


4. 绑定成功后，界面显示公网地址：



已有弹性公网 IP 绑定到 MySQL

1. 购买实例后，在 **实例管理** 页面，选择目标实例，单击实例名称，进入实例的基本信息页面。在左侧导航栏，单击连接管理，在连接信息 > 公网地址处，单击 **绑定**。



2. 在弹出框的弹性公网 IP 地址列表中，选择目标弹性公网 IP，单击**确定**，提交绑定任务。绑定成功后，界面显示公网地址：



接入配置

步骤1：创建测试数据库

在 **实例管理** 页面，选择目标实例，单击**实例名称**，进入实例的基本信息页面。在左侧导航栏，单击**数据库管理 > 创建数据库**，创建测试数据库。

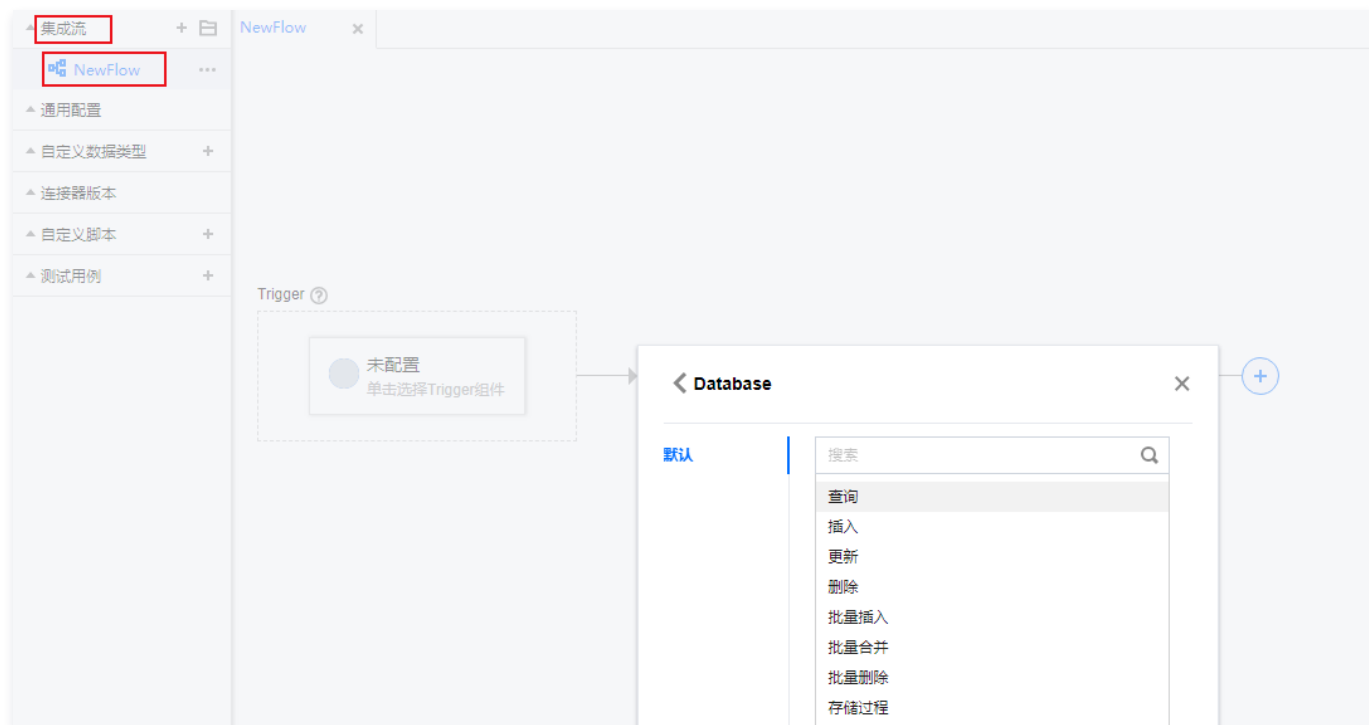


步骤2：配置腾讯云数据连接器 Database 连接器连接属性

1. 在 **腾讯云数据连接器控制台**，单击**添加应用**，选择空白应用进行创建。



2. 单击左侧的集成流展开选择 **NewFlow**，在画布中单击 + 选择 Database 连接器相关操作。



3. 单击右侧的新建连接器配置并按照指引填入相关参数。



4. 填写连接器配置参数后，单击**测试连接**，测试连通性，测试成功后，会出现“连接配置正确”的提示，保存连接器配置即可，失败根据提示来重新填写对应信息。

- 数据库类型：此处选择 MySQL
- 地址：填写绑定的弹性公网 IP，可参见 [弹性公网IP信息](#) 进行填写
- 端口号：填写数据库访问端口号，可参见 [访问端口号](#) 进行填写
- 数据库：填写 [步骤1](#) 中创建的测试数据库名称
- 用户名：填写购买时设置的用户名称，可参见 [设置数据库密码](#)

○ 密码：填写购买时设置的用户密码，可参见 [设置数据库密码](#)

新建连接器配置

选择连接器

2 配置连接器

连接配置

数据库类型

MySQL

地址 *

填写绑定的弹性公网IP

数据库地址

端口号 *

int 3306

填写数据库访问端口号

端口号范围：0-65535

数据库 *

test

填写数据库名称

数据库库名

用户名 *

root

填写购买时设置的用户

账户名称

密码 *

填写用户名密码

账户密码

编码格式

utf8

高级配置

超时时间(秒)

int 5

最长超时时间：30秒

安全网关

安全网关

无

测试连接

上一步

保存

5. 其他相关配置可参见 [Database 连接器使用指南](#)。

腾讯云 MySQL 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:01

操作场景

Database 连接器支持 MySQL 数据库的连接及增删改查等数据库常用操作。

前期准备

您需要先开通 MySQL，并已配置好安全组规则和外网访问域名。

购买云数据库 MySQL

登录 [腾讯云数据库 MySQL](#) 购买页面，根据需求购买云数据库。详细操作可参见 [购买方式](#)。

注意

默认内网端口为3306，可自定义，后面安全组的端口配置需根据此处选择的端口号填写（外网端口映射至内网端口进行访问）。

设置安全组

设置允许腾讯云数据连接器出口 IP 访问数据库的操作。详细操作可参见 [MySQL 安全组配置](#)。

新建安全组

1. 单击 [安全组](#) 或购买页面的新建，进入安全组创建页面，模板选择自定义或放通全部端口。

新建安全组

×

模板

自定义

名称

放通22, 80, 443, 3389端口和ICMP协议
放通全部端口
自定义

所属项目

默认项目

备注

自定义

高级选项

显示模板规则

确定

取消

2. 单击配置规则（选择自定义需操作此步骤，上一步选择全部端口放通单击确定即可）。

提醒

×

建议您立即设置安全组规则，因为无规则的安全组将禁止所有流量，可能导致您的服务器无法登陆或访问内外网。

立即设置规则

取消

3. 添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	3306（若数据库自定义端口填写自定义端口号）	0.0.0.0/0	开启通过公网访问 MySQL

 **注意**

若需配置指定的腾讯云数据连接器出口IP访问，请 [提交工单](#) 联系我们。

修改已有安全组

1. 在 [安全组实例列表](#) 中，单击要修改的安全组实例右侧的修改规则，对已有规则进行修改。

安全组  所有项目

安全组帮助文档 

新建 编辑标签

多个关键字用逗号分隔

ID/名称	关联实例数	备注	类型	更新时间	创建时间	项目	操作
 安全组实例名称-2022...	0	暴露全部端口到公网...	自定义	2022-04-11 16:43:52	2022-04-11 16:43:51	默认项目	修改规则 管理实例 更多
 自定义-2022041116...	0	自定义	自定义	2022-04-11 16:43:35	2022-04-11 16:43:35	默认项目	修改规则 管理实例 更多
 安全组实例名称	0	-	自定义	2020-10-20 16:56:47	2020-10-20 16:56:47	默认项目	修改规则 管理实例 更多
 安全组实例名称	0	-	自定义	2020-09-14 11:05:35	2020-09-14 11:05:35	默认项目	修改规则 管理实例 更多
 安全组实例名称	0	默认	默认	2019-09-18 10:46:59	2019-09-18 10:46:59	默认项目	修改规则 管理实例 更多

2. 修改入方向规则，添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	3306（若数据库自定义端口填写自定义端口号）	0.0.0.0/0	开启通过公网访问 MySQL

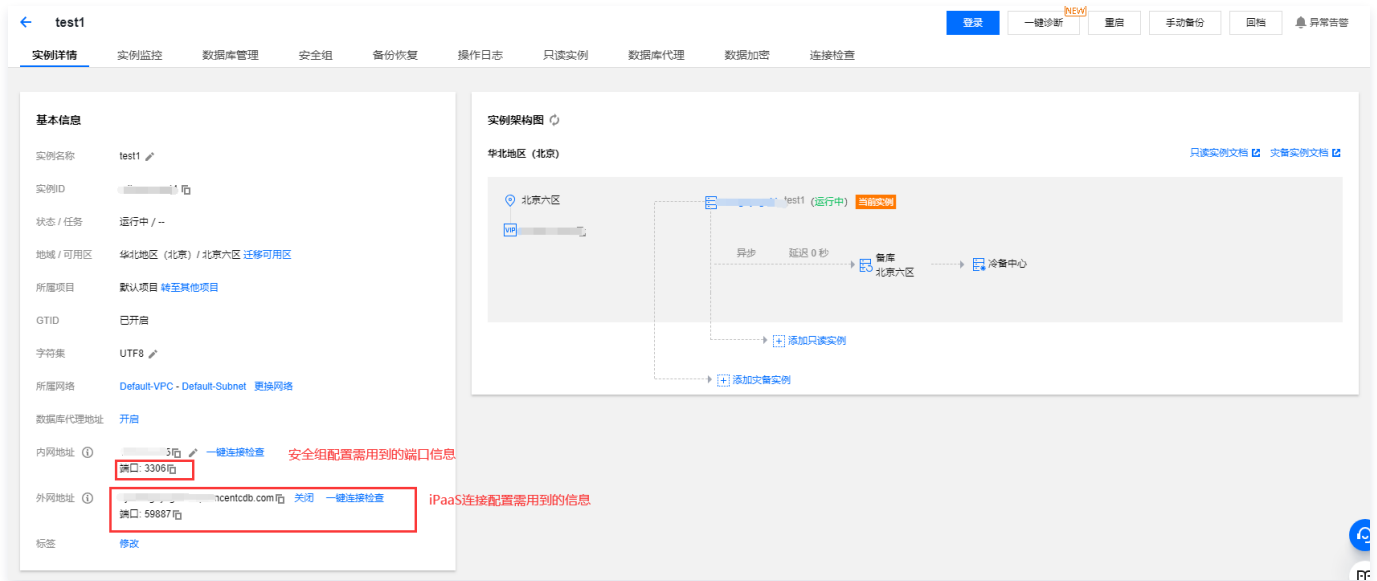
 **注意**

若需配置指定的腾讯云数据连接器出口 IP 访问，请 [提交工单](#) 联系我们。

开启外网地址访问

1. 购买成功后，登录 [云数据库 MySQL 控制台](#)。
2. 在实例列表中选中想要设置的实例，进入实例管理页面。

3. 单击**开启**，开启外网地址访问。



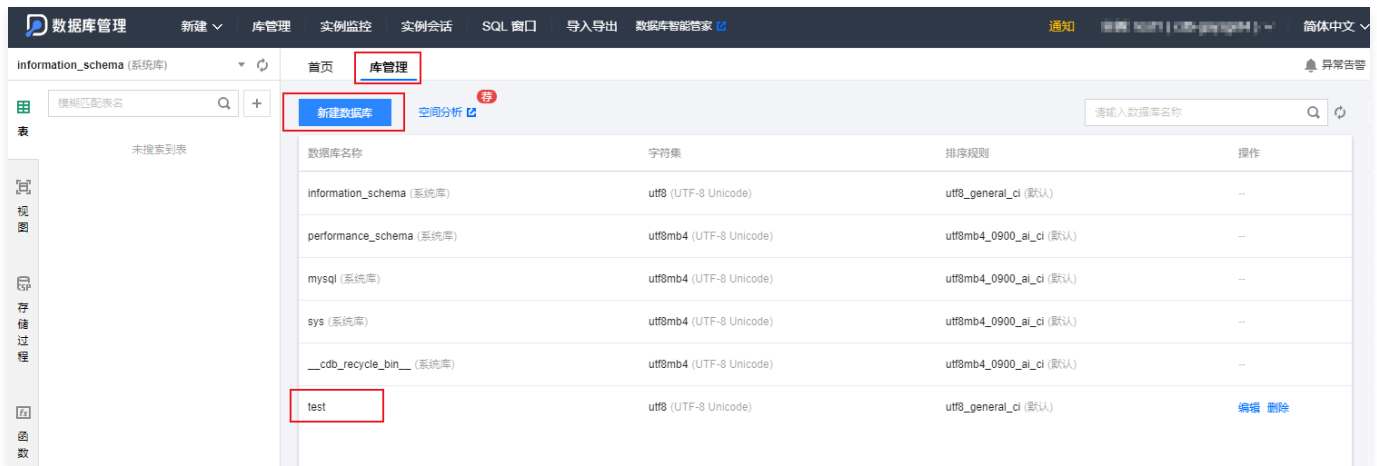
接入配置

步骤1: 创建测试数据库

1. 进入数据库管理页面，单击**创建数据库**。输入数据库的登录账密后进入库管理页面再次单击**创建数据库**。

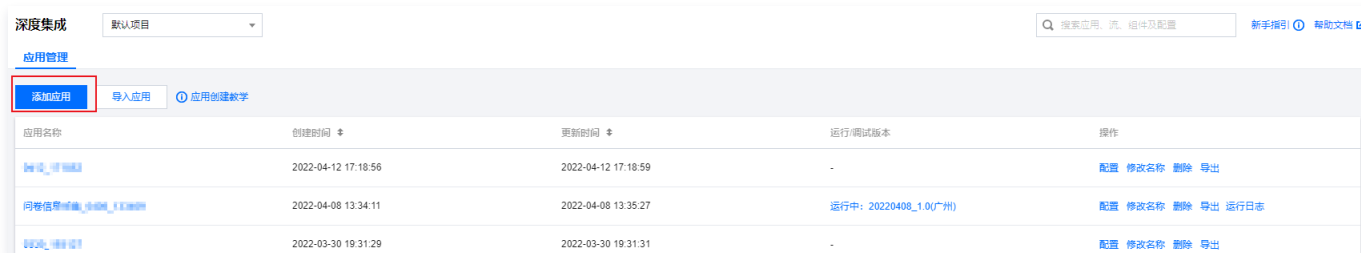


2. 输入数据库的登录账密后进入库管理页面单击**新建数据库**。

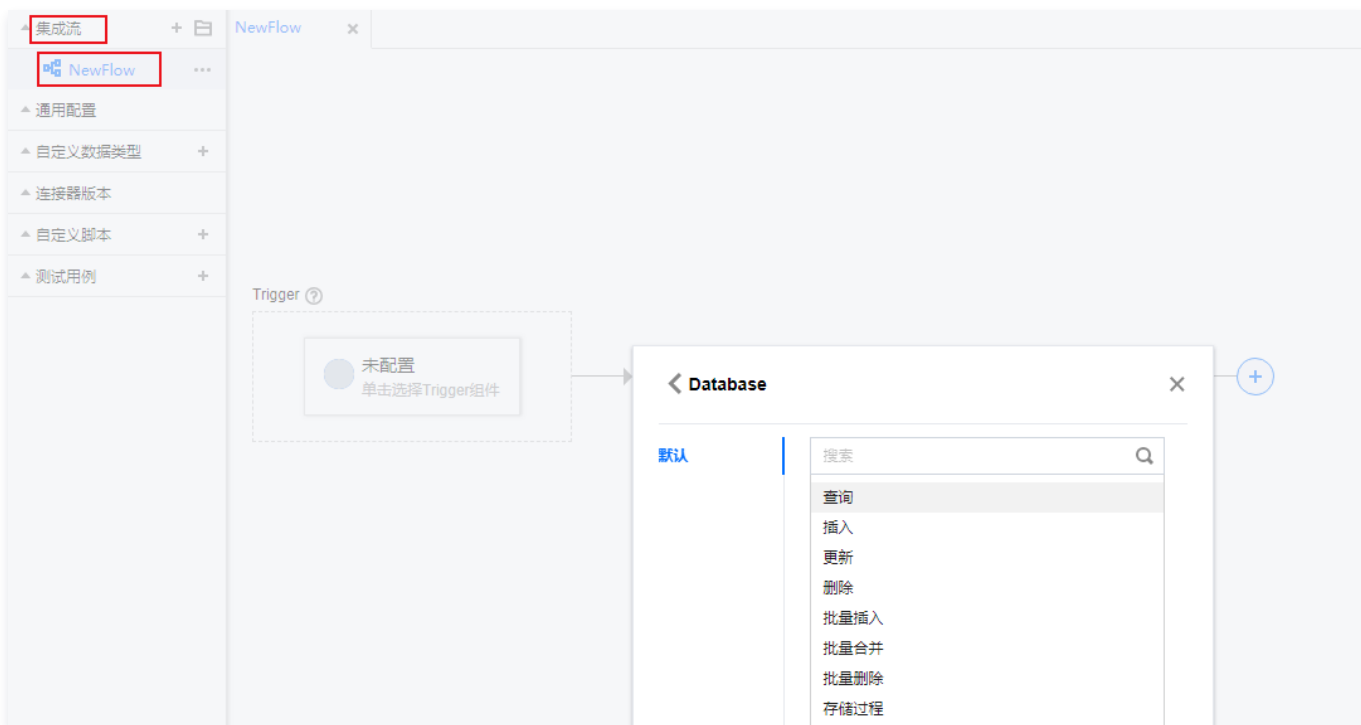


步骤2：配置腾讯云数据连接器 Database 连接器连接属性

1. 在 [腾讯云数据连接器控制台](#)，单击**新建应用**，选择**空白应用**进行创建。



2. 单击左侧的**集成流**展开选择 **NewFlow** 在画布中单击 + 选择 **Database** 连接器相关操作。



3. 单击右侧的**新建连接器配置**并按照指引填入相关参数。



4. 填写连接器配置参数后，单击**测试连接**，测试连通性，测试成功后，会出现“连接配置正确”的提示，保存连接器配置即可，失败根据提示来重新填写对应信息。

- 数据库类型：此处选择 MySQL
- 地址：填写数据库外网访问域名，可参见 [开启外网地址访问](#) 中的外网地址进行填写
- 端口号：填写数据库外网访问端口号，可参见 [开启外网地址访问](#) 中的端口号进行填写
- 数据库：填写 [步骤1](#) 中创建的测试数据库名称
- 用户名：填写购买时设置的用户名称，可参见 [购买云数据库 MySQL](#)

○ 密码：填写购买时设置的用户密码，，可参见 [购买云数据库 MySQL](#)

新建连接器配置

✓ 选择连接器

>

2 配置连接器

连接配置

数据库类型

MySQL

地址 *

bj-1.tencentcdb.com

填写数据库外网访问域名

数据库地址

端口号 *

int 59887

填写数据库外网访问端口号

端口号范围: 0-65535

数据库 *

test

填写数据库名称

数据库库名

用户名 *

root

填写数据库用户名

账户名称

密码 *

填写数据库用户密码

账户密码

编码格式

utf8

高级配置

超时时间(秒)

int 5

最长超时时间: 30秒

安全网关

安全网关

无

测试连接

上一步

保存

5. 其他相关配置可参见 [Database 连接器使用指南](#)。

Redis 连接器

Redis 连接器使用指南

最近更新时间：2023-06-08 14:48:01

简介

Redis 是使用 ANSI C 编写的开源、支持网络、基于内存、分布式的 [键值对存储数据库](#)。

Redis 连接器可连接第三方 Redis 服务并执行操作。用户通过连接器配置来配置 Redis 的连接参数，配置成功后便可执行 Redis 操作。

连接器配置

连接方式

Redis 连接器支持三种 Redis 服务的连接方式：

- Cluster 模式
- NonCluster 模式
- Sentinel 模式

配置参数

Redis 连接器配置参数如下：

参数	数据类型	描述	是否必填	默认值
Redis 连接模式	enum	Redis 连接模式，支持 Cluster 模式、NonCluster 模式、Sentinel 模式	是	—
集群地址	string	Cluster 模式参数，Redis 集群地址，格式为 host:port，以逗号相隔	是	—
地址	string	NonCluster 模式参数，Redis 服务地址	是	—
端口号	int	NonCluster 模式参数，Redis 服务端口号	是	—
哨兵地址集合	string	Sentinel 模式参数，哨兵地址集合，格式为 host:port，以逗号相隔	是	—
主节点名称	string	Sentinel 模式参数，哨兵集群主节点名称	否	—
密码	string	密码，用于验证 Redis 连接	否	—

配置界面

Cluster 模式连接器配置界面：

通用

连接配置

Redis连接模式

Cluster模式

集群地址

string

127.0.0.1:6379,127.0.0.1:6479

f(x)

格式为host:port,以逗号相隔

密码

用于验证Redis连接

NonCluster 模式连接器配置界面：

通用

连接配置

Redis连接模式

NonCluster模式

地址

127.0.0.1

Redis服务地址

端口号

int

请输入整数

端口号范围：0-65535

密码

用于验证Redis连接

Sentinel 模式连接器配置界面：

通用

连接配置

Redis连接模式

Sentinel模式

哨兵地址集合

string

127.0.0.1:6379,127.0.0.1:6479

f(x)

格式为host:port,以逗号相隔

主节点名称

mymaster

密码

用于验证Redis连接

操作说明

Redis 连接器的操作分为键操作、字符串、列表、哈希、集合、有序集合等操作。

Redis

键操作

字符串

列表

哈希

集合

有序集合

搜索

exists

del

expire/pexpire

expireat/pexpireat

ttl/pttl

persist

键操作

exists

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 int 类型，1表示键存在，0表示键不存在；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。



2. 新建连接器配置，填写配置参数，单击测试连接，测试连接器配置是否正确。

编辑连接器配置

✓ 链接配置正确

✓ 选择连接器

>

2 配置连接器

通用

连接配置

Redis连接模式

NonCluster模式

地址

9.139.43.102

Redis服务地址

端口号

int

6379

端口号范围：0-65535

密码

.....

用于验证Redis连接

高级配置

安全网关

无

测试连接

上一步

保存

3. 在通用配置中，填入待查询的键。

Redis exists

当前使用的是 redis 物理配置

编辑 | 切换

配置

调试

备注

通用

键

string

test_key

f(x)

4. 执行成功后，message payload 中包含查询结果。

输出消息

payload

▼

0

del

删除键。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 int 类型，1表示键删除成功，0表示删除的键不存在；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入待删除的键。
4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。

expire/pexpire

设置键的过期时间。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
过期时间	int	键的过期时间	是	—
时间单位	enum	过期时间的单位，支持秒、毫秒	是	秒

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 bool 类型，true 表示键过期时间设置成功，false 表示键不存在或者不能为键设置过期时间；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? expireat/pe...

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

通用

键 *

string

test_key

f(x)

键的过期时刻 *

int

def dw_process(msg): return ms

🗑

uinx时间戳

时间戳单位

毫秒

- 4. 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。上例执行成功，返回 true。

输出消息 payload ▾

true

expireat/pexpireat

设置键的过期时刻。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
键的过期时刻	int	键的过期时刻，时间戳格式	是	—

时间戳单位	enum	过期时刻的时间戳单位，支持秒、毫秒	是	秒
-------	------	-------------------	---	---

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 bool 类型，true 表示键过期时刻设置成功，false 表示设置失败；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? expireat/pe...

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼

test_key

f(x)

键的过期时刻 *

int ▼

def dw_process(msg): return ms

🗑

uinx时间戳

时间戳单位

毫秒 ▼

- 4. 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。上例执行成功，返回 true。

输出消息 payload ▼

true

ttd/pttd

获取键的剩余生存时间。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
时间单位	enum	键剩余生存时间的时间单位，支持秒、毫秒	是	秒

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 int 类型，-2表示键不存在，-1表示键存在但没有设置剩余生存时间，其他情况表示按时间单位返回的键剩余生存时间；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 添加 Redis 连接器组件。
- 新建连接器或选择已创建的连接器。
- 在通用配置中，填写相关参数。例如：

Redis ? ttl/pttl

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string

test_key

f(x)

时间单位

秒

- 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。上例执行成功，按秒为单位，返回键的剩余生存时间。

输出消息 payload

99999512

persist

移除键的过期时间。

参数配置

参数	数据类型	描述	是否必填	默认值
----	------	----	------	-----

键	string	键	是	-
---	--------	---	---	---

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 bool 类型，true 表示键的过期时间移除成功，false 表示键不存在或键没有设置过期时间；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 添加 Redis 连接器组件。
- 新建连接器或选择已创建的连接器。
- 在通用配置中，填写相关参数。例如：

Redis ? persist

当前使用的是 redis 物理配置

编辑 | 切换

配置

调试

备注

通用

键

string

test_key

f(x)

- 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。上例执行成功，返回 true。

输出消息 payload

true

字符串操作

get				
获取键的值。				
参数配置				
参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 string 类型，表示键对应的值；当键不存在或执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? get

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键 *

string

test_key

f(x)

- 4. 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。
 - 上例执行成功，返回键值：

输出消息 payload

"test"

- 若键不是字符串类型，返回错误：

输出消息 error

{
 "Code": "CORE:RUNTIME",
 "Description": "WRONGTYPE Operation against a key holding the wrong kind of value"
}

set/setnx

设置键的值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
值	string	值	是	-
键的超时时间	enum	键的超时时间，单位为秒，默认为0，不设置超时时间	否	0
键存在时使操作无效	bool	开启对应 Redis SETNX 操作，关闭对应 Redis SET 操作	否	false

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，键设置成功后，payload 值为"OK"，设置失败后，payload 值为 false；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? set/setnx

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼

test-key

f(x)

值 *

string ▼

ipaas

f(x)

键的超时时间
(单位:秒)

int ▼

0

f(x)

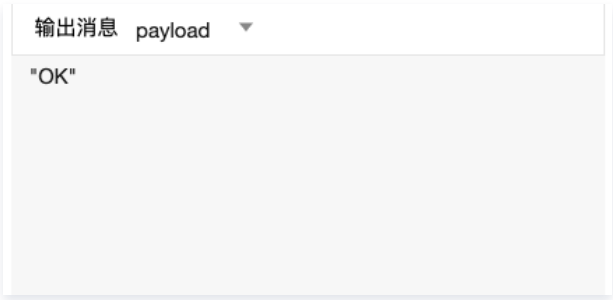
当键存在时使操作无效

bool ▼

false ▼

开启对应Redis SETNX操作， 关闭对应Redis SET操作

4. 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。上例执行成功：



incrby/incrbyfloat

将键存储的值加上增量值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
增量值	int/float	增量值	是	-
数值类型	enum	增量值的数值类型，支持整型、浮点型	是	1

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为执行命令后键对应的值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? incrbymincr...

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼

test_key

f(x)

增量值

string ▼

1

f(x)

数值类型

浮点型 ▼

4. 执行成功后，message payload 中保存执行结果；执行失败后，message error 保存报错信息。

○ 上例执行成功：

payload ▼ ⓘ

2

○ 若键不是数值类型，报错：

error ▼ ⓘ

```
{
  "Code": "CORE:RUNTIME",
  "Description": "ERR value is not an integer or out of range"
}
```

decrby

将键存储的值减去减量值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
减量值	int	减量值	是	1

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 值为执行命令后键对应的值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。
4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。

getrange

获取键的子字符串。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
起始索引	int	字符串的起始索引值	是	—
终止索引	int	字符串的终止索引值	是	—

输出

操作执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为截取得到的子字符串；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? getrange

...

⌵

当前使用的是 redis 物理配置

编辑 | 切换

配置

调试

备注

通用 ▲

键 *

string ▼

test-key

📄

f(x)

起始索引 *

int ▼

0

f(x)

终止索引 *

int ▼

2

f(x)

⚠ 注意

test-key 对应的字符串为"ipaas"。

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，返回截取后的字符串：

payload ▼ ⓘ

"ipa"

strlen

获取键字符串的长度。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 int 类型，表示键字符串的长度；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。

2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? strlen

...

☐

当前使用的是 redis 物理配置

编辑 | 切换

配置 调试 备注

通用 ▲

键

string ▼ test-key f(x)

⚠ 注意

test-key 对应的字符串为"ipaas"。

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，返回截取后的字符串：

payload ▼ ⓘ

5

列表操作

push-to-list

将元素插入列表。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
待插入值	string	待插入元素值	是	-
列表左端插入	bool	开启表示由列表左端插入，关闭表示由列表右端插入	否	false
列表不存在时是否操作无效	bool	开启表示当键不存在时操作无效，关闭表示当键不存在时创建新的列表	否	false

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 int 类型，表示插入元素后列表的长度；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? push-to-list

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼

test-list

f(x)

列表对应的键

待插入值 *

string ▼

1

f(x)

列表左端插入

bool ▼

true

▼

开启表示由列表左端插入，关闭表示由列表右端插入

列表不存在时是否操作无效

bool ▼

false

▼

开启对应Redis Lpushx/Rpushx操作，关闭对应Redis Lpush/Rpush操作

- 4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。

pop-from-list

移除列表元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
移除列表左端元素	bool	开启表示移除列表左端元素，关闭表示移除列表右端元素	否	false

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值表示列表移除的元素值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? pop-from-list

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键 *

string

test-list

f(x)

列表对应的键

移除列表左端元素

bool

false

开启对应Redis Lpop操作，关闭对应Redis Rpop操作

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，返回移除的元素值：

payload

"1"

llen

返回列表的长度。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 int 类型，表示列表的长度；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。

4. 执行成功后，message payload 中保存查询结果；执行失败后，Message error 保存报错信息。

lindex

通过索引获取列表元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
索引值	int	列表的索引值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值表示列表中下标为指定索引值的元素；如果指定索引值不在列表的区间范围内或执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? lindex

...

⌵

当前使用的是 redis 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用 ▲

键 *

string ▼

test-list



f(x)

列表对应的键

索引值 *

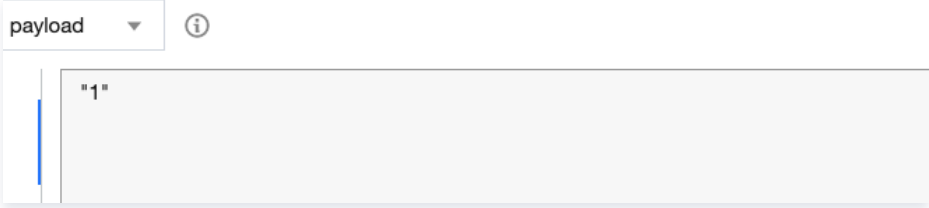
int ▼

1



f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：



lset

通过索引设置元素的值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
索引值	int	列表的索引值	是	-
值	string	待设置的值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为"OK"；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? hset

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼ test-list  f(x)

列表对应的键

索引值 *

int ▼ 1 f(x)

待设置值 *

string ▼ 333 f(x)

4. 执行成功后，message payload中保存了查询结果；执行失败后，message error保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

"OK"

哈希操作

hset

为哈希表字段赋值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
字段名	string	哈希表的字段名	是	-
值	string	哈希表字段对应的值	是	-
哈希字段名不存在时是否操作无效	bool	开启对应 Redis HSETNX 操作，关闭对应 Redis HSET 操作	否	false

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 bool 类型，true 表示设置成功，false 表示哈希表中字段已经存在且旧值已被新值覆盖或给定字段已经存在且没有操作被执行；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息

variable	继承上个组件的 variable 信息
----------	---------------------

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? hset

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼

test-hash

f(x)

哈希字段名 *

string ▼

aaa

f(x)

值 *

string ▼

111

f(x)

哈希字段名不存在时是否操作无效

bool ▼

false

▼

开启对应Redis HSETNX操作，关闭对应Redis HSET操作

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

true

hgetall

获取哈希表的所有字段和值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 dict 类型，键为哈希表字段名，值为哈希表字段值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息

attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  {
    "id": 1,
    "name": "a",
    "price": "25.16"
  },
  {
    "id": 2,
    "name": "b",
    "price": "15.26"
  }
]
```

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? hgetall

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用 ▲

键 *

string ▼

test-hash

f(x)

哈希对应的键

- 4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

```
{
  "fff": "111",
  "test1": "aaa",
  "test2": "bbb"
}
```

hget

获取哈希表指定字段的值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
哈希字段名	string	哈希表字段名称	是	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为哈希表字段对应的值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis hget

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键

string

test-hash

f(x)

哈希对应的键

哈希字段名

string

aaa

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ⓘ

"1"

hincrby/hincrbyfloat

为哈希表字段值加上增量值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
哈希字段名	string	哈希表字段名称	是	-
增量值	int/float	增量值	是	1
数值类型	enum	增量值的类型，支持整型、浮点型	是	整型

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为操作执行后字段的值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? hincrbby/hin...

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

通用

键 *

string ▼

test-hash

f(x)

哈希对应的键

哈希的字段名 *

string ▼

test3

f(x)

增量值

string ▼

1

f(x)

数值类型

整型 ▼

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

2

hlen

为哈希表字段值加上增量值。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 int 类型，表示哈希表字段的数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? hlen

当前使用的是 redis 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

键 *

string

test-hash

f(x)

哈希对应的键

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

4

hdel

删除哈希表字段。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
字段名	string	哈希表字段名称	是	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 值为哈希表成功删除字段的数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis hdel

当前使用的是 redis 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

键

string

test-hash

f(x)

哈希对应的键

哈希字段名

string

test1

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

1

集合操作

sadd

集合添加元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
元素值	string	待添加的元素值	是	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 值为插入成功的元素数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? sadd

当前使用的是 redis 物理配置 编辑 | 切换

配置

调试

备注

通用

键 *

string

test-set

f(x)

集合对应的键

元素值 *

string

1

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

1

spop

移除集合元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为移除的元素值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息

attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? spop

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键 *

string

test-set

f(x)

集合对应的键

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

"1"

srandmember

返回集合随机元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为元素值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? srandmember

当前使用的是 redis 物理配置 编辑 | 切换

配置

调试

备注

通用

键 *

string

test-set

f(x)

集合对应的键

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

i

"2"

sismember

判断元素是否是集合元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
元素值	string	元素值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 bool 类型，true 表示元素在集合存在，false 表示元素不存在；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? sismember

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键 *

string

test-set

f(x)

集合对应的键

元素值 *

string

2

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

ⓘ

true

scard

返回集合元素数量。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为集合元素数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? scard

...

当前使用的是 redis 物理配置

编辑 | 切换

配置

调试

备注

通用

▲

键 *

string ▼

test-set

f(x)

集合对应的键

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼

?

2

srem

移除集合指定元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
元素值	string	待移除的元素值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为成功删除的元素数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

Redis ? srem

当前使用的是 redis 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

键 *

string ▼

test-set

f(x)

集合对应的键

元素值 *

string ▼

2

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

1

有序集合操作

zadd

返回有序集合元素数量。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	—
元素值	string	待添加的元素值	是	—
元素分数	float	元素值对应的分数	是	—

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为成功添加的新成员的数量，不包括被更新的、已经存在的成员；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Redis 连接器组件。

2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

Redis ? zadd

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼ test-zset f(x)

有序集合对应的键

元素值 *

string ▼ test1 f(x)

元素分数 *

string ▼ 100 f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

1

zrange/zrevrange

返回有序集合指定区间的元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
起始索引	int	有序集合的起始索引	是	-
终止索引	int	有序集合的终止索引	是	-
升序显示元素值	bool	开启表示按元素分数值递增（从小到大）显示，关闭表示按元素分数值递减显示	是	true

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值 list 类型，表示有序集合的成员列表；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  "test1",
  "test2",
  "test3"
]
```

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? zrange/zrev...

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼ test-zset f(x)

有序集合对应的键

起始索引 *

int ▼ 0 f(x)

终止索引 *

int ▼ 2 f(x)

升序显示元素值

bool ▼ true ▼

开启对应Redis ZRANGE操作，关闭对应Redis ZREVRANGE操作

- 4. 执行成功后，message payload 中保存查询结果；执行失败后，Message error 保存报错信息。上例执行成功，结果如下：

payload ▼ ⓘ

```
[
  "test1",
  "test2",
  "test3"
]
```

zrangebyscore/zrevrangebyscore

返回有序集合中指定分数区间的成员列表。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
最小分数值	int	元素的最小分数值	是	-

最大分数值	int	元素的最大分数值	是	-
升序显示元素值	bool	开启表示按元素分数值递增（从小到大）显示，关闭表示按元素分数值递减显示	是	true

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值 list 类型，表示有序集合的成员列表；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  "test1",
  "test2",
  "test3".
]
```

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ? zrangebysc...

当前使用的是 redis 物理配置 编辑 | 切换

配置 调试 备注

通用

键 *

string ▼ test-zset f(x)

有序集合对应的键

最小分数值 *

int ▼ 0 f(x)

最大分数值 *

int ▼ 400 f(x)

升序显示元素值

bool ▼ true ▼

开启对应Redis ZRANGEBYSCORE操作，关闭对应Redis ZREVRANGEBYSCORE操作

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload ⓘ

```
[
  "test1",
  "test2",
  "test3"
]
```

zcard

返回有序集合元素数量。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
最小分数值	int	元素的最小分数值	是	-
最大分数值	int	元素的最大分数值	是	-
升序显示元素值	bool	开启表示按元素分数值递增（从小到大）显示，关闭表示按元素分数值递减显示	是	true

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为有序集合成员的数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

Redis ⓘ zcard ...

当前使用的是 redis 物理配置 编辑 | 切换

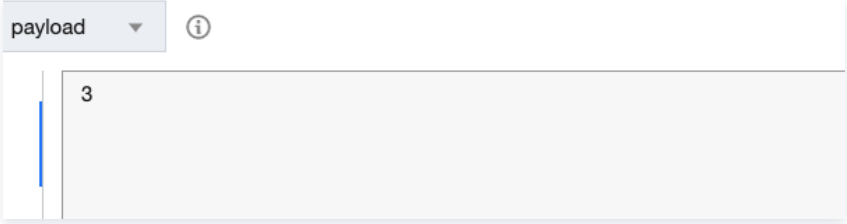
配置 调试 备注

通用 ▲

键 * string test-zset f(x)

有序集合对应的键

4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：



zrem

移除有序集合指定元素。

参数配置

参数	数据类型	描述	是否必填	默认值
键	string	键	是	-
元素值	int	元素值	是	-

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为有序集合成功移除的成员数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 Redis 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：



4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，结果如下：

payload

i

1	
---	--

阿里云 Redis 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:01

操作场景

Redis 连接器支持 Redis 单点模式、哨兵模式、Cluster 集群模式的连接及常用操作。
本文以 Redis 单点模式为例，为您介绍主流云厂商 Redis 产品的接入流程。

前期准备

您需要开通 Redis，并配置好外网访问规则和公网访问连接地址。

购买云数据库 Redis 实例

根据业务需求 [购买云数据库 Redis](#)，并设置对应的管理员账号。

地域和可用区

中国

亚太

欧洲与美洲

中东与印度

华东1（杭州）

华北1（青岛）

华北2（北京）

华南1（深圳）

华东2（上海）

华北3（张家口）

华北5（呼和浩特）

西南1（成都）

中国（香港）

华南2（河源）

华北6（乌兰察布）

华南3（广州）

可用区类型

单可用区

双可用区

【产品调整中，请先忽略可用区类型选项】主备节点都会部署在单一可用区，适合对可用区灾备级别有普通要求的场景。

主可用区

杭州 可用区 K

网络类型

专有网络

经典网络

如要通过ECS访问实例，请选择与ECS相同的VPC，实例一旦创建VPC不可更改。

专有网络

默认vpc

如果没有可用的专有网络，请前往控制台进行创建：[点击前往](#)

虚拟交换机

请选择

如果没有可用的交换机，请前往控制台进行创建：[点击前往](#)

版本类型

社区版

企业版

社区版产品，即开即用，简单上手，完全兼容 Redis 协议，[了解更多](#)。
欢迎选购企业版性能增强型，Redis全兼容，更强大性能，更多企业级能力，[了解二者对比](#)

版本号

Redis 5.0

Redis 4.0

[查看支持的命令](#)

架构类型

集群版

标准版

读写分离版

标准的 master-replica 架构，完全兼容 Redis 协议，[了解详情](#)

开通公网访问

1. 登录 [云数据库 Redis 控制台](#)（注意地域选择），选择目标实例，单击实例名称，进入实例的基本信息页面。在连接信息 > 公网访问处，单击申请连接地址。

实例信息

性能监控

报警设置

白名单设置

参数设置

账号管理

备份与恢复

服务可用性

日志管理

CloudDBA

基本信息

实例ID

状态

网络类型

专有网络

版本

实例规格

实例释放保护

带宽弹性伸缩

资源组名称

复制

运行中

VPC网络

vpc-

Redis 4.0 社区版

256M主从版

未开启 设置

未开启 修改

default resource group

实例名称

可用区

可维护时间段

交换机

小版本

最大连接数

最大私网带宽

资源组ID

修改

杭州 可用区K

02:00-06:00

vsw-

redis-4.0_1.9.7

10,000

10 MB/s 修改

rg-

迁移可用区

大版本升级

连接信息

账号管理

设置免密访问

连接类型

连接地址

端口号 (Port)

操作

专有网络

rds.redis.aliyuncs.com

6379

修改连接地址

公网访问

申请连接地址

* 提示：请使用以上访问连接串进行实例连接。VIP在业务维护中可能会变化。

2. 单击**确认提交申请**。

申请连接地址

连接类型：

公网访问

连接地址：

r-

.redis.rds.aliyuncs.com

端口号：

6379

!

端口号修改生效大约需要10分钟，请生效后刷新页面查看最新端口号信息

3. 配置成功后，控制台页面会显示公网连接地址及端口号。后续填写在腾讯云数据连接器 Redis 连接器配置中。

← r- 运行中

登录数据库 迁移Redis数据库 清除数据 变更配置 更多 刷新

实例信息

性能监控 报警设置 白名单设置 参数设置 账号管理 备份与恢复 服务可用性 日志管理 CloudDBA

基本信息

实例ID

r- 复制

状态

运行中

网络类型

VPC网络

专有网络

vpc- 修改

版本

Redis 4.0 社区版

实例规格

256M主从版

实例释放保护

未开启 设置

带宽弹性伸缩

未开启 修改

资源组名称

default resource group

实例名称

修改

可用区

杭州 可用区K

可维护时间段

02:00-06:00 1 修改

交换机

vsw- 修改

小版本

redis-4.0.19.7

最大连接数

10,000

最大私网带宽

10 MB/s 修改

资源组ID

rg-

连接信息

账号管理 设置免密访问

连接类型	连接地址	端口号 (Port)	操作
专有网络	r-  .redis.rds.aliyuncs.com 复制	6379	修改连接地址
公网访问	r-  .redis.rds.aliyuncs.com 复制	6379	释放连接地址 修改连接地址

* 提示： 请使用以上访问连接串进行实例连接，VIP在业务维护中可能会变化。

设置访问白名单

登录 [云数据库Redis控制台](#)（注意地域选择），选择目标实例，单击**实例名称**，进入实例的基本信息页面。在左侧导航中单击**白名单设置** > **添加白名单分组设置** 0.0.0.0/0 允许被访问。

!

注意

若需配置指定的腾讯云数据连接器 出口 IP 访问，请 [提交工单](#) 联系我们。

添加白名单分组

* 分组名称

redis1

* 添加方式

手动添加

加载ECS私网IP

* 组内白名单

0.0.0.0/0

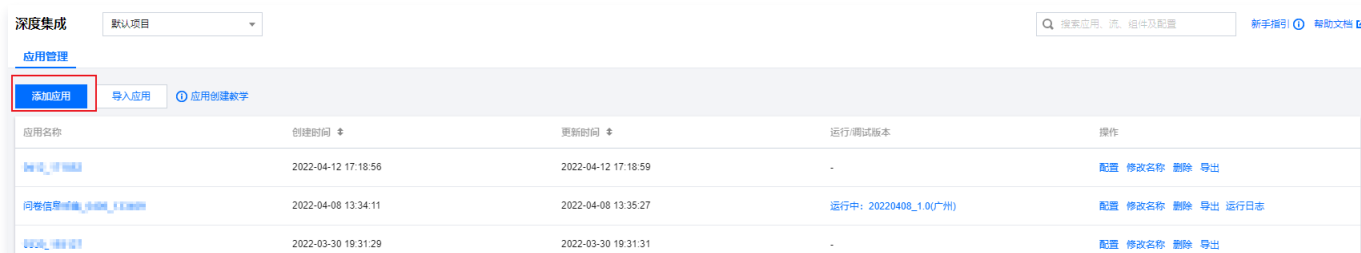
确定

取消

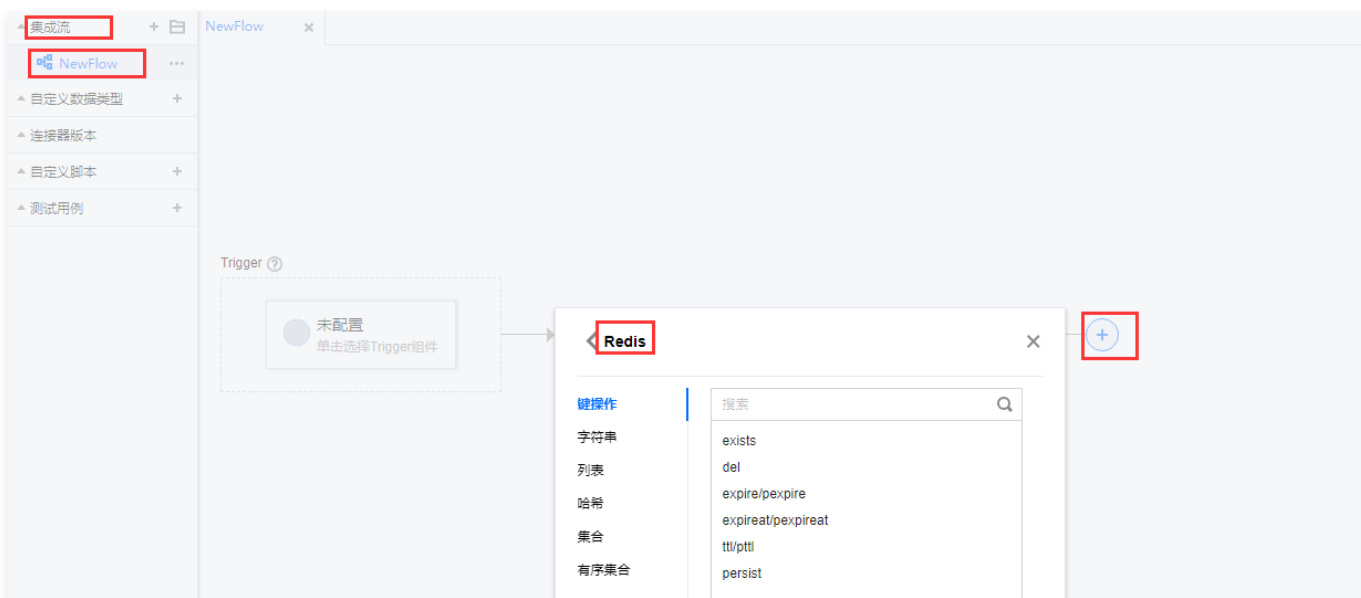
接入配置

配置腾讯云数据连接器 Database 连接器连接属性步骤如下：

1. 在 [腾讯云数据连接器控制台](#)，单击新建应用，选择空白应用进行创建。



2. 单击左侧的集成流展开选择 NewFlow 在画布中单击 + 选择 Redis 连接器相关操作。



3. 单击右侧的新建连接器配置并按照指引填入相关参数。



4. 填写完连接器配置参数后，单击测试连接，测试连通性，测试成功后，会出现“连接配置正确”的提示，保存连接器配置即可，失败根据提示来重新填写对应信息。

- Redis 连接模式：此处选择 Noncluster 模式
- 地址：填写 [外网访问域名](#)
- 端口号：填写 [数据库访问端口](#)
- 密码：填写 [购买数据库时的密码](#)

编辑连接器配置

✕

✓ 连接配置正确

通用

▲

连接配置

Redis连接模式NonCluster模式

地址r-redis-1234567890.redis.rds.aliyuncs.com

Redis服务地址

端口号int6379

端口号范围: 0-65535

密码*****

用于验证Redis连接

高级配置

▲

安全网关无

▼

测试连接

保存

5. 其他相关配置可参见 [Redis 连接器使用指南](#)。

最近更新时间: 2024-11-25 10:11:12

Redis 连接器支持 Redis 单点模式、哨兵模式、Cluster 集群模式的连接及常用操作。本文以 Redis 单点模式为例，介绍主流云厂商 Redis 产品的接入流程。

您需要先开通 Redis，并配置好外网访问规则和公网访问。

1. 购买 分布式缓存服务 Redis。

2. 创建实例时，提示“当前版本不支持安全组，请创建完成后配置白名单”，因此无需绑定安全组。

虚拟私有云

default_vpc

default_subnet (/24)

子网DNS服务器地址配置不正确，会导致实例域名无法解析。关于如何切换内网DNS，请参考帮助中心的[如何切换内网DNS](#)。

如需创建新的虚拟私有云，可前往[控制台](#)创建。[了解DCS实例和虚拟私有云、子网的关系](#)

IP地址

自动分配IP地址

/

使用默认端口：6379

安全组

当前版本不支持安全组，请创建完成后配置白名单

访问方式

密码访问

免密访问

密码

.....

请妥善保管密码，系统无法获取您设置的密码内容。

确认密码

.....

参数配置

系统默认

使用自定义模板

数量

-

1

+

您还可以购买DCS缓存实例160个，使用64,000 GB内存。[申请扩大配额](#)

配置外网访问

购买云服务器

❗ DCS 实例支持公网访问吗？

- 华为云 DCS-Redis 只有3.0实例支持公网访问，Redis4.0 和 Redis5.0 实例不支持绑定弹性 IP，无法通过公网访问，但是华为云官网目前 Redis3.0 已经下架，只能购买 Redis4.0 和 Redis5.0 实例且腾讯云数据连接器处于华为云的外部需要外网访问，所以这里采用 Nginx 实现公网访问 Redis 4.0/5.0 实例的访问，也可参见华为云文档 [使用 Nginx 实现公网访问 Redis 4.0/5.0 的单机/主备/Proxy 集群实例](#)。
- [华为云常见问题 >>](#)

- 购买与华为云 Redis 处于同一可用区同一 VPC 下内网互通的有外网的云服务器。
可参见 [华为云 ECS 购买指引](#)。
- 登录服务器，部署 Nginx。

安装 Nginx

❗ 说明：

本文以云服务器操作系统为 Centos7.x 为例进行安装，不同操作系统命令稍有不同。

- 执行以下命令，添加 Nginx 到 yum 源。

```
sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7ngx.noarch.rpm
```

- 添加完之后，执行以下命令，查看是否已经添加成功。

```
yum search nginx
```

- 添加成功之后，执行以下命令，安装 Nginx。

```
sudo yum install -y nginx
```

4. 执行以下命令安装 stream 模块。

```
yum install nginx-mod-stream --skip-broken
```

5. 启动 Nginx 并设置为开机自动运行。

```
sudo systemctl start nginx.service
sudo systemctl enable nginx.service
```

```
[root@ecs-77f0 ~]# sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.nginx.noarch.rpm
Retrieving http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.nginx.noarch.rpm
Warning: /var/tmp/rpm-tmp.Zgq46: Header V4 RSA/SHA1 Signature, key ID 7bd9bf62: NOKEY
Preparing...##### [100%]
Updating / installing...
 1:nginx-release-centos-7-0.el7.nginx##### [100%]
[root@ecs-77f0 ~]#
[root@ecs-77f0 ~]# sudo yum install -y nginx
Loaded plugins: fastestmirror
Determining fastest mirrors
base                                                    | 3.6 kB  00:00:00
epel                                                    | 4.7 kB  00:00:00
extras                                                  | 2.9 kB  00:00:00
nginx                                                   | 2.9 kB  00:00:00
updates                                                 | 2.9 kB  00:00:00
(1/8): base/7/x86_64/group_gz                          | 153 kB  00:00:00
(2/8): epel/x86_64/group_gz                            | 96 kB  00:00:00
(3/8): epel/x86_64/updateinfo                          | 1.1 MB  00:00:00
(4/8): epel/x86_64/primary_db                          | 7.0 MB  00:00:00
(5/8): base/7/x86_64/primary_db                        | 6.1 MB  00:00:00
(6/8): extras/7/x86_64/primary_db                      | 246 kB  00:00:00
(7/8): updates/7/x86_64/primary_db                     | 14 MB  00:00:00
(8/8): nginx/x86_64/primary_db                         | 71 kB  00:00:02
Resolving Dependencies
--> Running transaction check
--> Package nginx.x86_64 1:1.20.2-1.el7.nginx will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
nginx x86_64 1:1.20.2-1.el7.nginx nginx 790 k
Transaction Summary
=====
Install 1 Package
Total download size: 790 k
Installed size: 2.8 M
Downloading packages:
nginx-1.20.2-1.el7.nginx.x86_64.rpm | 790 kB 00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
Installing : 1:nginx-1.20.2-1.el7.nginx.x86_64
1/1

Thanks for using nginx!

Please find the official documentation for nginx here:
* https://nginx.org/en/docs/

Please subscribe to nginx-announce mailing list to get
the most important news about nginx:
* https://nginx.org/en/support.html
```

6. 在本地浏览器中输入服务器地址（公网 IP 地址），查看安装是否成功。如果出现下面页面，则表示安装成功。

```
[root@ecs-77f0 ~]# sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.nginx.noarch.rpm
Retrieving http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.nginx.noarch.rpm
Warning: /var/tmp/rpm-tmp.Zgq46: Header V4 RSA/SHA1 Signature, key ID 7bd9bf62: NOKEY
Preparing...##### [100%]
Updating / installing...
 1:nginx-release-centos-7-0.el7.nginx##### [100%]
[root@ecs-77f0 ~]#
[root@ecs-77f0 ~]# sudo yum install -y nginx
Loaded plugins: fastestmirror
Determining fastest mirrors
base                                                    | 3.6 kB  00:00:00
epel                                                    | 4.7 kB  00:00:00
extras                                                  | 2.9 kB  00:00:00
nginx                                                   | 2.9 kB  00:00:00
updates                                                 | 2.9 kB  00:00:00
(1/8): base/7/x86_64/group_gz                          | 153 kB  00:00:00
(2/8): epel/x86_64/group_gz                            | 96 kB  00:00:00
(3/8): epel/x86_64/updateinfo                          | 1.1 MB  00:00:00
(4/8): epel/x86_64/primary_db                          | 7.0 MB  00:00:00
(5/8): base/7/x86_64/primary_db                        | 6.1 MB  00:00:00
(6/8): extras/7/x86_64/primary_db                      | 246 kB  00:00:00
(7/8): updates/7/x86_64/primary_db                     | 14 MB  00:00:00
(8/8): nginx/x86_64/primary_db                         | 71 kB  00:00:02
Resolving Dependencies
--> Running transaction check
--> Package nginx.x86_64 1:1.20.2-1.el7.nginx will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package Arch Version Repository Size
=====
Installing:
nginx x86_64 1:1.20.2-1.el7.nginx nginx 790 k
Transaction Summary
=====
Install 1 Package
Total download size: 790 k
Installed size: 2.8 M
Downloading packages:
nginx-1.20.2-1.el7.nginx.x86_64.rpm | 790 kB 00:00:01
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
Warning: RPMDB altered outside of yum.
Installing : 1:nginx-1.20.2-1.el7.nginx.x86_64
1/1

Thanks for using nginx!

Please find the official documentation for nginx here:
* https://nginx.org/en/docs/

Please subscribe to nginx-announce mailing list to get
the most important news about nginx:
* https://nginx.org/en/support.html
```

修改 Nginx 配置文件

1. 打开并修改配置文件。

① 说明：

proxy_pass 参数配置值为同一 VPC 下的 Redis 实例的 IP 地址，具体可从缓存实例详情页面的“连接信息”区域获取。

执行 `vi /etc/nginx.conf` 打开配置文件。

配置示例如下，在 proxy_pass 中配置 Redis 实例连接地址。

```
stream {
    server {
        listen 8080;
        proxy_pass 192.168.0.6:6379;
    }
    server {
        listen 8081;
        proxy_pass 192.168.0.6:6379;
    }
}
```

2. 重启 Nginx，并查看 Nginx 端口状态。

```
[root@ecs-77f0 ~]# service nginx restart
Redirecting to /bin/systemctl restart nginx.service
[root@ecs-77f0 ~]# netstat -an|grep 808
tcp        0      0 0.0.0.0:8080        0.0.0.0:*          LISTEN
tcp        0      0 0.0.0.0:8081        0.0.0.0:*          LISTEN
tcp        0  478      0 192.168.0.19:8080  113.108.77.61:6693 ESTABLISHED
```

设置安全组允许腾讯云数据连接器访问

在 [ECS 实例控制台](#)，修改 ECS 安全组配置，配置允许腾讯云数据连接器访问。

⚠ 注意：

若需配置指定的腾讯云数据连接器出口 IP 访问，请 [提交工单](#) 联系我们。

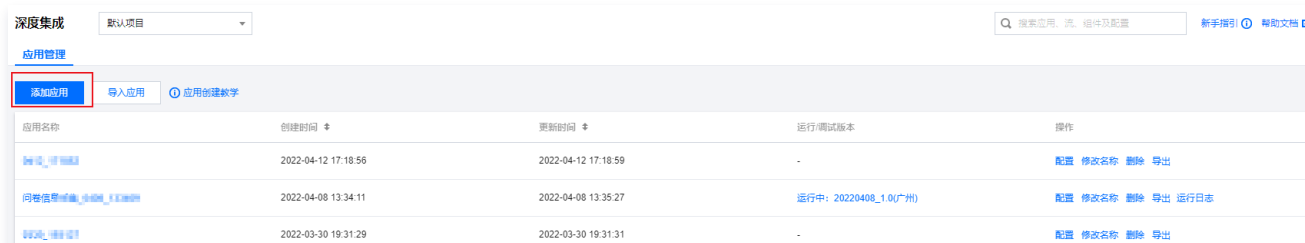
The screenshot shows the 'Security Group' configuration page in the Tencent Cloud console. On the left, there's a list of security groups: 'Sys-FullAccess', 'default', and 'Sys-WebServer'. The 'Sys-FullAccess' group is selected. On the right, the 'Inbound Rules' (入方向规则) are listed. A table shows the rules for 'Sys-FullAccess':

安全组名称	优先级	策略	协议端口	类型	源地址	描述
Sys-FullAccess	1	允许	全部	IPv6	Sys-FullAccess	--
	1	允许	全部	IPv4	0.0.0.0/0	--
	1	允许	全部	IPv4	Sys-FullAccess	--
	1	允许	全部	IPv6	::/0	--

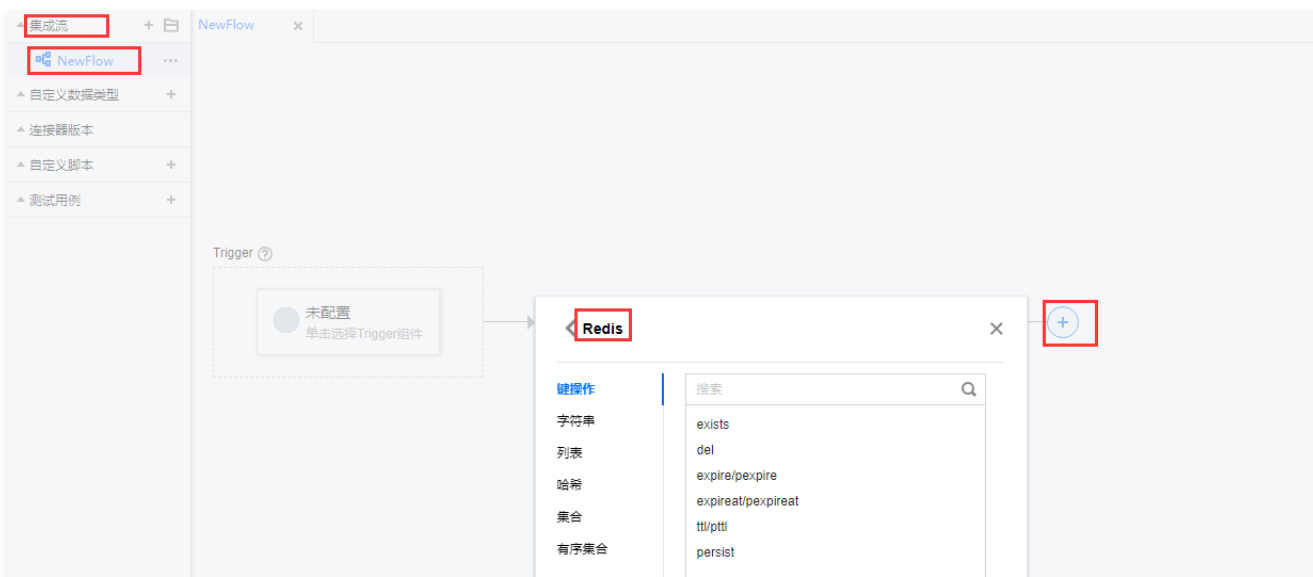
接入配置

配置腾讯云数据连接器 Database 连接器连接属性的步骤如下：

1. 在 [腾讯云数据连接器控制台](#)，单击新建应用，选择空白应用进行创建。



2. 单击左侧的集成流展开选择 NewFlow 在画布中单击 +选择 Redis 连接器相关操作。



3. 单击右侧的新建连接器配置并按照指引填入相关参数。



4. 填写完连接器配置参数后，单击测试连接，测试连通性，测试成功后，会出现连接配置正确的提示，保存连接器配置即可，失败根据提示来重新填写对应信息。

- Redis 连接模式：此处选择 Noncluster 模式。
- 地址：填写 配置外网访问中创建的 ECS 的外网 IP。
- 端口号：填写 修改 Nginx 配置文件 中 proxy_pass 设置的 listen 端口。
- 密码：填写购买数据库时的密码。

编辑连接器配置

✕

✔ 连接配置正确

通用

▲

连接配置

Redis连接模式NonCluster模式▼

地址

Redis服务地址

端口号

int ▼

8080

端口号范围: 0-65535

密码

用于验证Redis连接

高级配置

▲

安全网关无▼

测试连接

保存

5. 其他相关配置可参见 [Redis 连接器使用指南](#)。

腾讯云 Redis 接入腾讯云数据连接器

最近更新时间：2023-06-08 14:48:01

操作场景

Redis 连接器支持 Redis 单点模式、哨兵模式、Cluster 集群模式的连接及常用操作。本文以 Redis 单点模式为例，介绍主流云厂商 Redis 产品的接入流程。

前期准备

您需要先开通 Redis 服务，并配置好外网访问规则和公网访问连接地址。

购买云数据库 Redis

⚠ 注意

网络类型选择“私有网络”，“基础网络”不能开启公网地址访问。

登录 [腾讯云数据库 Redis 购买页面](#)，根据需求购买云数据库并设置密码。详细操作可参见 [购买方式](#)。

设置安全组

设置允许腾讯云数据连接器出口 IP 访问数据库的操作。详细操作可参见 [Redis 配置安全组](#)。

新建安全组

1. 单击 [安全组](#) 或购买页面的[自定义安全组](#)，进入安全组创建页面。

当前网络选择下, 仅“to”网络的主机可访问数据库. [新建私有网络](#) [新建子网](#)

端口

自定义端口号需在1024到65535之间

参数模板

Redis 4.0 标准架构默认模板
ID: default-param-template-6



[新建参数模板](#)

指定项目

默认项目



[新建项目](#)

标签 ②

标签键

标签值

操作

[添加](#)

如现有的标签不合适, 您可以去控制台[新建标签/标签值](#)

安全组 ②

[选择已有安全组](#)

[使用指引](#)

未选择数据



已选择安全组 (共0条)

如您有业务需要放通其他端口, 您可以 [自定义安全组](#)

自研上云用户建议选择云梯默认安全组

2. 模板选择自定义或放通全部端口。

新建安全组

模板

自定义

名称

放通22, 80, 443, 3389端口和ICMP协议
放通全部端口
自定义

所属项目

默认项目

备注

自定义

高级选项

显示模板规则

确定

取消

3. 单击立即设置规则（选择自定义需操作此步骤，上一步选择全部端口放通单击确定即可）。

提醒

建议您立即设置安全组规则，因为无规则的安全组将禁止所有流量，可能导致您的服务器无法登陆或访问内外网。

立即设置规则

取消

4. 添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	ALL（若数据库自定义端口填写自定义端口号）	0.0.0.0/0	开启通过公网访问 redis

注意

若需配置指定的腾讯云数据连接器出口 IP 访问，请 [提交工单](#) 联系我们。

选择已有安全组后修改

1. 在 [安全组实例列表](#) 中，单击已绑定 Redis 数据库的安全组右侧的**修改规则**，对已有规则进行修改。

安全组

所有项目

安全组帮助文档

新建

编辑标签

多个关键字用竖线 | 分隔

☐	ID/名称	关联实例数	备注	类型	更新时间	创建时间	项目	操作
☐	安全组实例列表 放通全部端口-2022...	0	放通全部端口到公网...	自定义	2022-04-11 16:43:52	2022-04-11 16:43:51	默认项目	修改规则 管理实例 更多
☐	安全组实例列表 自定义-2022041116...	0	自定义	自定义	2022-04-11 16:43:35	2022-04-11 16:43:35	默认项目	修改规则 管理实例 更多
☐	安全组实例列表 安全组	0	-	自定义	2020-10-20 16:56:47	2020-10-20 16:56:47	默认项目	修改规则 管理实例 更多
☐	安全组实例列表 安全组	0	-	自定义	2020-09-14 11:05:35	2020-09-14 11:05:35	默认项目	修改规则 管理实例 更多
☐	安全组实例列表 安全组	0	安全组	默认	2019-09-18 10:46:59	2019-09-18 10:46:59	默认项目	修改规则 管理实例 更多

2. 修改入方向规则，添加如下入站规则。

方向	协议	端口	源地址	说明
入方向	TCP	ALL（若数据库自定义端口填写自定义端口号）	0.0.0.0/0	开启通过公网访问 redis

注意

若需配置指定的腾讯云数据连接器出口 IP 访问，请 [提交工单](#) 联系我们。

开启外网地址访问

1. 登录 [云数据库 Redis 实例控制台](#)，在实例列表选中想要设置的实例，进入实例管理页面，开启外网地址访问。

← crs-

实例详情

节点管理

系统监控

安全组

账号管理

参数配置

备份与恢复

慢查询

任务记录

基本信息

实例名称

crs-

实例ID

crs-

实例状态

运行中

可用区

广州六区

所属项目

默认项目

读写状态

读写

规格信息

产品版本

内存版

兼容版本

Redis 4.0

架构版本

标准架构

内存容量

1GB, 已用 34.89MB, (3.4%)

内存配置

1 分片/1GB/1 副本

副本只读

未开启

网络信息

所属网络

vpc-test

所在子网

subnet-test

内网IPv4地址

外网地址

开启

最大连接数

10,000 个

最大网络吞吐

128Mb/s

2. 配置成功后，控制台会显示外网地址。后续腾讯云数据连接器的连接器配置需填写对应字段。

实例详情

节点管理

系统监控

安全组

账号管理

参数配置

备份与恢复

慢查询

任务记录

基本信息

实例名称

crs-

实例ID

crs-

实例状态

运行中

可用区

广州六区

所属项目

默认项目

读写状态

读写

规格信息

产品版本

内存版

兼容版本

Redis 4.0

架构版本

标准架构

内存容量

1GB, 已用 34.89MB, (3.4%)

内存配置

1 分片/1GB/1 副本

副本只读

未开启

网络信息

所属网络

vpc-test

所在子网

subnet-test

内网IPv4地址

外网地址

com:27583 关闭

最大连接数

10,000 个

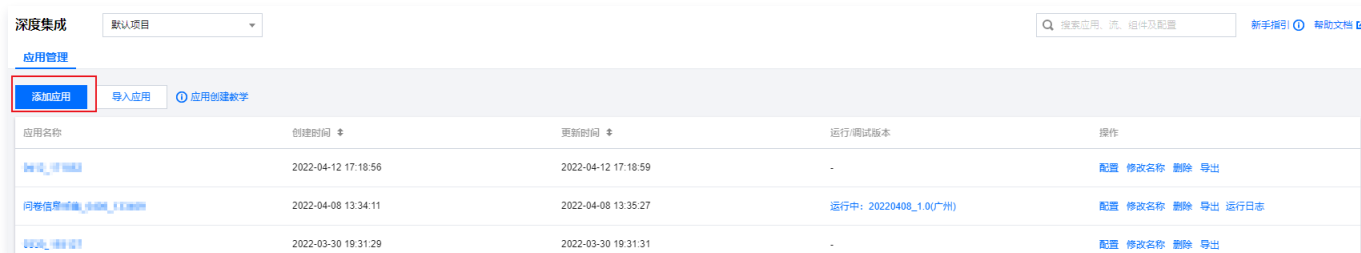
最大网络吞吐

128Mb/s

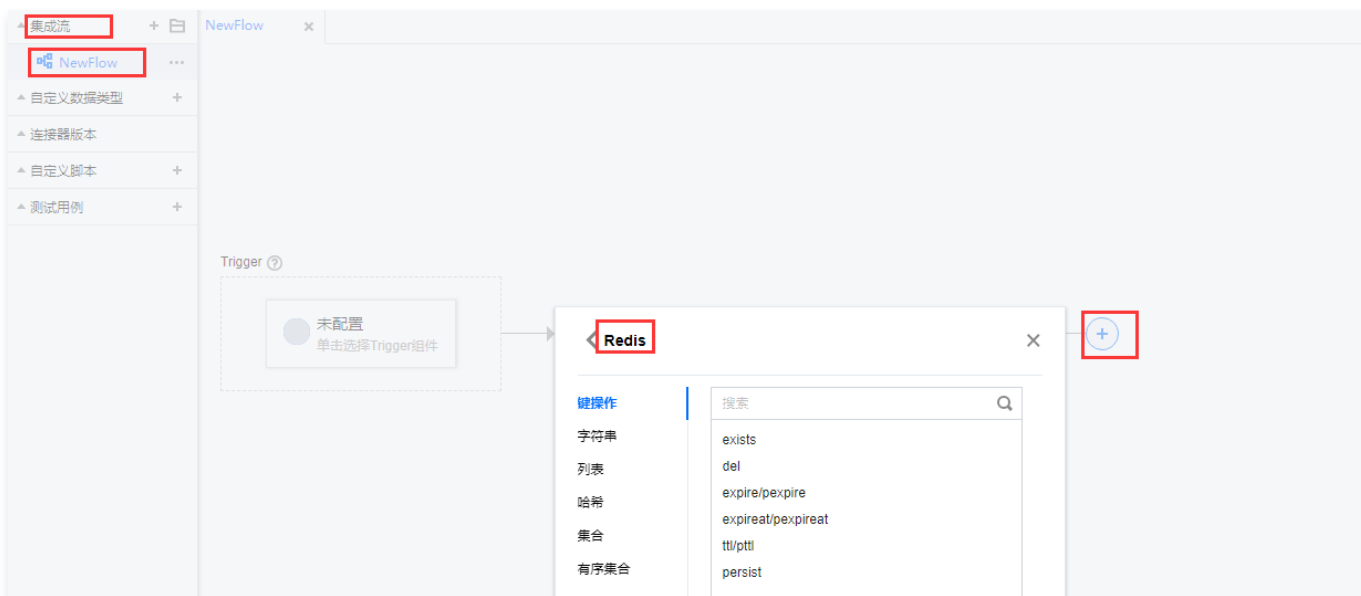
接入配置

配置腾讯云数据连接器 Database 连接器连接属性步骤如下：

1. 在 [腾讯云数据连接器控制台](#) 上单击**新建应用**，选择**空白应用**进行创建。



2. 单击左侧的**集成流**展开选择 **NewFlow** 在画布中单击 **+选择 Redis 连接器** 相关操作。



3. 单击右侧的**新建连接器配置**并按照指引填入相关参数。



4. 填写完连接器配置参数后，单击**测试连接**，测试连通性，测试成功后，会出现“连接配置正确”的提示，保存连接器配置即可，失败根据提示来重新填写对应信息。
 - Redis 连接模式：此处选择 Noncluster 模式
 - 地址：填写 [外网访问域名](#)
 - 端口号：填写 [数据库访问端口](#)
 - 密码：填写 [购买数据库时的密码](#)

5. 其他相关配置可参 [Redis 连接器使用指南](#)。

HTTP Listener 连接器

最近更新时间：2023-06-08 14:48:01

简介

HTTP Listener 连接器可以接收 HTTP 请求并生成消息，从而作为 Trigger 触发集成流。

连接器配置

- 基本配置：HTTP Listener 连接器基本配置中包括描述和监听域名两项，其中监听域名为必填项。

编辑连接器配置

选择连接器

配置连接器

基本配置

高级配置

描述

请输入连接器描述信息

监听域名 *

80

运行时后缀

上一步

保存

- 高级配置：HTTP Listener 连接器高级配置中包括 IP 白名单和认证配置两项。

编辑连接器配置

选择连接器

配置连接器

基本配置

高级配置

IP白名单

公网IP白名单, 以换行分隔

认证配置 ?

账号

密码

操作

请输入用户名

请输入密码

删除

+ 添加账号

上一步

保存

- 上述连接器配置参数相关描述如下表所示：

参数	数据类型	描述	是否必填	默认值
监听域名	string	Listener 的监听域名，只支持由大小写字母、数字组成，不得超过15个字符。此处指定的为域名前缀，完整的域名取决于部署集群的配置，例如：配置域名为 prefix，则发布之后域名为 prefix-\$appId.ipaas.\$region.myqcloud.com，应用测试域名为 prefix-\$appId.ipaas.sandbox.myqcloud.com，其中 \$appId 为当前用户 appId，\$region 表示发布的地域，例如：发布地域选择“天津”时，\$region 为“ap-tianjin”	是	无
IP 白名单	CIRD/点分十进制字符	客户端 IP 白名单限制	否	无

	串			
认证配置	账号&密码字符串	HTTP 身份认证，默认 Basic 模式	否	无

操作说明

HTTP Listener 操作配置包括基本配置、高级配置、响应配置三项：

Http Listener ?

...

□

当前使用的是 listener-config 物理配置

编辑 | 切换

配置 备注

基本配置

监听路径 *
/
指定监听的路径

监听方法 *
全部
指定监听HTTP请求的方法

高级配置

输出消息绑定自定义数据类型

消息属性
payload

类型选择
Custom_Type
+

数据消费模式
repeatable ⓘ

响应配置

正常响应

状态码
int 请输入整数 f(x)
正常响应时返回的状态码

响应头
def dw_process(msg): return
正常响应时的HTTP头，仅支持dict和multimap数据结构，表达式返回值规范请参考Abcabcd或Abc-Abc

响应体
def dw_process(msg): return
正常响应时返回的内容

错误响应

状态码
int 请输入整数 f(x)
错误响应时返回的状态码

响应头
def dw_process(msg): return
错误响应时的HTTP头，仅支持dict和multimap数据结构，表达式返回值规范请参考Abcabcd或Abc-Abc

响应体
def dw_process(msg): return
错误响应时返回的内容

基础配置参数描述

参数	参数类型	描述	是否必填	默认值
----	------	----	------	-----

监听 路径	string	Listener 通过监听路径区分不同的 URI，同时支持三种模式路径匹配：通配符 *、变量匹配、精确匹配。例如：监听路径设置为 /user/{userId}/* 时，可以匹配 /user/123/testpath，也可以匹配 /user/123 和 /user/123/testpath/testpath1，即 {} 可以匹配一级目录，*** 匹配多级目录。当多个请求同时匹配一个模式时，会优先匹配更具体的，同等条件下，优先匹配第一个。	是	“/”
请求 方法	string	下拉框多选，全集为：全部、GET、POST、PUT、PATCH、DELETE、HEAD、OPTIONS	是	全部

高级配置参数描述

参数	参数 类型	描述	是否 必填	默认值
消息属 性	string	下拉框选择输出消息绑定的属性，HTTP Listener 只支持绑定 payload	否	payload
类型选 择	string	下拉框选择输出消息 payload 绑定的数据类型，包括 message 内置数据类型和自定义数据类型	否	空
数据消 费模式	string	下拉框选择是否对消息数据进行持久化处理： repeatable：默认处理方式，消息数据可在一次触发中持续重复使用 non-repeatable：数据一次性消费，用于大文件数据的流式高效处理，暂时支持 JSON 数组类型，其超过 5M 时建议选择该模式	否	repeatable

响应配置参数描述

正常响应/错误响应	参数类型	描述	是否必填	默认值
状态码	int/表达式	响应的 HTTP 状态码	否	正常：200，错误：500
响应头	表达式，仅支持 dict 和 multimap	响应的 HTTP 头	否	无
响应体	表达式	响应的 body 体	否	正常：msg.payload，错误：msg.error

输出

HTTP Listener 生成的消息中，会将接收到请求的基本信息、Header、Query 参数以及 Path 参数保存在消息的 attributes；接收到请求的 body 保存在消息的 payload；执行失败后，错误信息会保存在生成消息的 error。

HTTP Listener 输出消息描述：

消息属性	值
payload	请求 body 保存在 payload 中，支持所有内置消息类型和自定义消息类型，其中 application/json、application/xml、application/yaml、application/x-www-form-urlencoded、multipart/form-data 等结构的 payload 支持在表达式中结构化访问
error	保存集成流执行结果中的错误，执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attributes	保存接收到请求的属性值，具体如下个表格所示
variables	默认为空

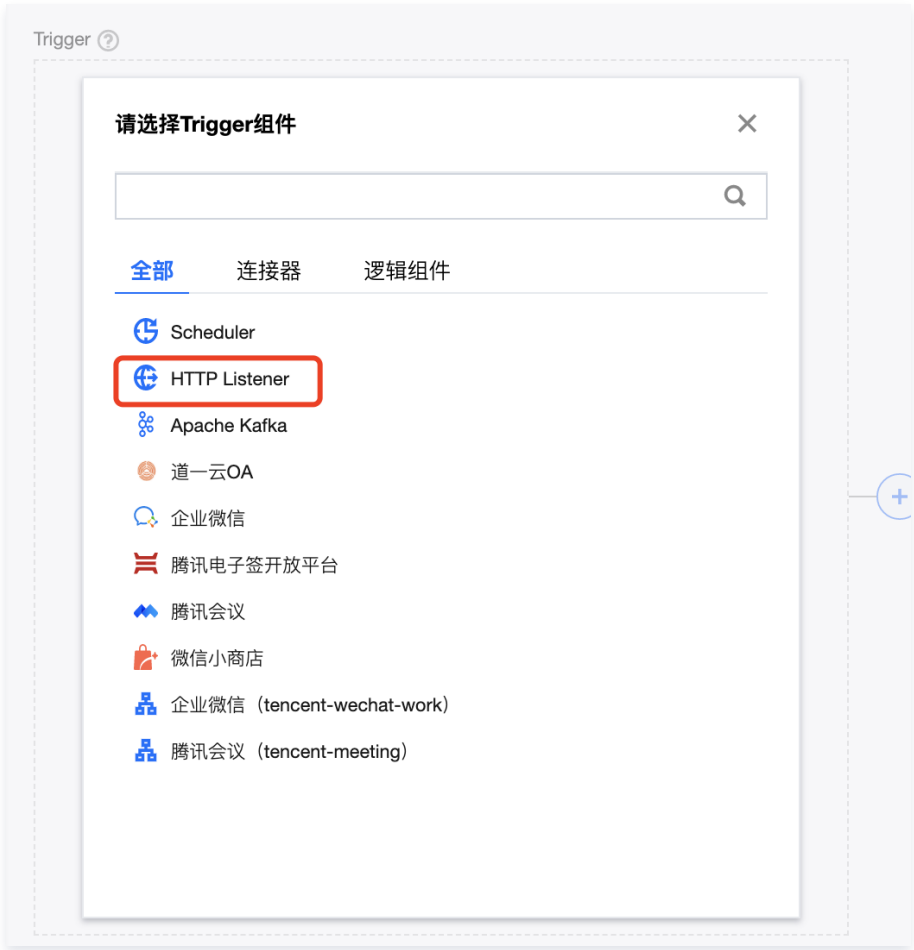
attributes 接收请求属性描述：

属性	描述
header	请求的 header，结构为：dict<string,list>

cookies	请求的 cookies，结构为：dict<string,string>，解析 Cookies 逻辑比较常见，可通过该数据直接读取
host	请求的 Host，例如：xxx-\$appld.ipaas.sadbox.qcloud.com
method	请求的 method，例如：GET、POST
requestUri	请求的原始 URI，包括 query 参数，例如：/api/v1/foo? sss=bbb
requestPath	请求的 Path，例如：/api/v1/foo
remoteAddr	请求是来源 IP，例如：127.0.0.1:1111
queryParams	请求的 query 参数，即 URI 中 “?” 后面的参数，结构为 dict<string,list>
uriParams	请求的 path 参数，当监听路径中包含 {varName} 变量时，可以从该参数中获取，结构为 dict<string,string>

案例

1. Trigger 配置选择 HTTP Listener。



2. 新建并配置 HTTP Listener 连接器。

3. 配置必填项监听路径和监听方法。

4. 发布并触发。

使用 POST 请求对应的 Listener，将请求 body 设置成 JSON 格式的 {"key":"value"}，HTTP Listener 会默认响应 body 为消息的 payload，同时也是接收请求的 body：

- 应用测试模式下，可以清楚看到 payload 内容：

版权所有：腾讯云计算（北京）有限责任公司

attributes:

输出消息 attributes ▾

```
{
  "cookies": {},
  "headers": {
    "Accept": [
      "*"/*
    ],
    "Accept-Encoding": [
      "gzip, deflate, br"
    ],
    "Cache-Control": [
      "max-age=259200"
    ],
    "Connection": [
      "keep-alive"
    ],
    "Content-Length": [
      "21"
    ],
    "Content-Type": [
      "application/json"
    ],
    "Postman-Token": [
      "2a00958c-777f-44c6-9ba2-99bdc76ad802"
    ],
    "User-Agent": [
      "PostmanRuntime/7.26.8"
    ],
    "X-Client-Proto": [
      "http"
    ],
    "X-Client-Proto-Ver": [
      "HTTP/1.1"
    ],
    "X-Forwarded-For": [
      "113.108.77.70"
    ],
    "X-Forwarded-Proto": [
      "http"
    ],
    "X-Real-Ip": [
      "113.108.77.70"
    ],
    "X-Stgw-Time": [
      "1619410781.030"
    ]
  },
  "host": "80-1302414305.ipaas.sandbox.myqcloud.com",
  "method": "POST",
  "queryParams": {},
  "remoteAddr": "9.165.192.65:37339",
  "requestPath": "base/path1/path2",
  "requestUri": "/base/path1/path2",
  "uriParams": {
    "test": [
      "path1"
    ]
  }
}
```

附录

HTTP Listener 错误描述如下:

错误码	错误类型	错误描述
401	HTTP:LISTENER_AUTHENTICATION_REQUIRED	身份认证异常
403	HTTP:LISTENER_CLIENT_IP_NOT_ALLOWED	请求 IP 白名单异常
404	HTTP:LISTENER_NOT_FOUND	请求 URL 错误
500	HTTP:SYSTEM_ERROR	系统异常
500	HTTP:LISTENER_MODULE_ERROR	获取模块失败

500	HTTP:LISTENER_RESOURCE_LIMIT_ERROR	资源管控
500	HTTP:LISTENER_OUT_OF_QUOTA	频率限制
500	HTTP:LISTENER_CONFIG_REF_ERROR	获取连接器配置异常
500	HTTP:LISTENER_HEADERS_EVAL_ERROR	响应配置 Header 表达式异常
500	HTTP:LISTENER_HEADERS_TYPE_ERROR	响应配置 Header 类型错误
500	HTTP:LISTENER_SET_COOKIE_EVAL_ERROR	响应配置 Cookies 表达式异常
500	HTTP:LISTENER_BODY_EVAL_ERROR	响应配置 Body 表达式异常

HTTP Request 连接器

最近更新时间：2023-06-08 14:48:01

简介

HTTP Request 连接器作为 HTTP 客户端，可以发起 HTTP 请求并将得到的响应作为消息传递给下一个组件。

配置说明

根据您创建应用的时间不同，HTTP Request 连接器配置也不相同，具体如下：

配置方法	适用场景	配置入口
方法一	在2021年9月2日前创建的应用	在新建连接配置页面进行基本配置和高级配置
方法二	在2021年9月2日及之后创建的应用	直接在右侧弹出的配置界面中即可进行基本配置和高级配置
方法三	在2022年4月8日及之后创建的应用	配置连接 > 配置请求信息

与方案一相比，方案二和方案三的优化内容如下：

- 方法二优化内容：此项优化不影响原有和现有应用的配置和使用，只是在配置操作上进行了优化，更方便客户填写对应信息。
- 方法三优化内容：HTTP Request 连接器支持连接配置的统一管理。

方法一（2021年9月2日前创建的应用）

参数介绍

- 基本配置：HTTP Request 连接器基本配置中包括描述、请求域名、请求协议、请求端口、最大重定向次数以及公共请求路径，其中请求域名、请求协议、请求端口、最大重定向次数、公共请求路径为必填项。

编辑连接器配置

基本配置

高级配置

描述

请输入连接器描述信息

请求域名 *

jd.com

f(x)

请求协议 *

HTTPS

f(x)

请求端口 *

443

f(x)

最大重定向数 *

-

1

+

f(x)

公共请求路径 *

/

f(x)

保存

- 高级配置：HTTP Request 连接器高级配置中包括安全网关配置，用于内网域名/IP 请求。

编辑连接器配置

选择连接器

配置连接器

基本配置

高级配置

安全网关名称

无

上一步

保存

- 上述连接器配置参数相关描述如下表所示：

参数	数据类型	描述	是否必填	默认值
请求域名	string	对应的请求域名	是	无
请求协议	string	下拉框选择请求协议 HTTPS/HTTP	是	HTTPS
请求端口	string	请求端口，1-65535，默认443，与请求协议联动，HTTP 协议默认会是80	是	443
最大重定向数	int	重定向次数，最小值0，最大值1	是	1
公共请求路径	string	请求的公共路径，多次请求的路径中相同的部分可配置为公共请求路径	是	/
安全网关名称	string	直接请求内网域名/IP 时，需要绑定安全网关	否	无

配置说明

HTTP Request 操作配置包括基本配置、高级配置两项：

HTTP Request ?

...

☐

当前无可用连接器配置，需新建连接器配置以便后续调用

新建

配置

调试

备注

基本配置

▲

请求路径 *

string ▼

/

f(x)

指定发送请求的路径

请求方法 *

POST ▼

f(x)

指定发送请求的方法

请求头

def dw_process(msg): return

🗑

指定发送请求的请求头,支持表达式

请求参数

def dw_process(msg): return

🗑

指定发送请求的请求参数, 只支持表达式, 返回字典类型

请求体

string ▼

请输入字符串

f(x)

指定发送请求的请求体, 支持字面量和表达式

高级配置

▲

输出消息绑定自定义数据类型

消息属性

payload ▼

类型选择

空 ▼

+

数据消费模式

repeatable ▼

①

基础配置参数描述

参数	参数类型	描述	是否必填	默认值
请求路径	string/表达式	用于指定发送请求的路径	是	/
请求方法	string	下拉单选指定发送请求的方法，全集为：GET、POST、PUT、PATCH、DELETE、HEAD	是	GET
请求头	表达式	用于指定发送请求的请求头，结构为：dict<string,list>	否	无
请求参数	表达式	用于指定发送请求的请求参数，结构为：dict<string,string>	否	无
请求体	string/表达式	用于指定发送请求的请求体，非 GET、HEAD 方法是设置，支持字面量和表达式，默认是消息的 payload	否	无

高级配置参数描述

参数	参数类型	描述	是否	默认值
----	------	----	----	-----

			必填	
消息属性	string	下拉框选择输出消息绑定的属性，HTTP Request 只支持绑定 payload	否	payload
类型选择	string	下拉框选择输出消息 payload 绑定的数据类型，包括 message 内置数据类型和自定义数据类型	否	空
数据消费模式	string	下拉框选择是否对消息数据进行持久化处理： repeatable：默认处理方式，消息数据可在一次触发中持续重复使用 non-repeatable：数据一次性消费，用于大文件数据的流式高效处理，暂时支持 JSON 数组类型，其超过5M时建议选择该模式	否	repeatable

方法二（2021年9月2日及之后创建的应用）

参数介绍

- 基本配置：HTTP Request 连接器基本配置中包括请求 URL、请求协议、请求方法、请求头、请求参数、最大重定向次数以及请求超时时间，其中请求URL、请求方法、最大重定向次数为必填项。

- 高级配置：HTTP Request 连接器高级配置中输出消息绑定自定义数据类型，可用于自定义输出消息数据类型。绑定安全网关，用于内网域名/IP 请求。

HTTP Request

配置备注

基本配置

请求URL *

string

http://www.test.com

f(x)

指定发送请求的URL

请求方法 *

GET

f(x)

指定发送请求的方法

请求头

dict

点击构造dict

f(x)

指定发送请求的请求头,支持表达式、dict

请求参数

dict

点击构造dict

f(x)

指定发送请求的请求参数, 支持表达式、dict, 返回字典类型

最大重定向数 *

-

1

+

f(x)

最大重定向数

请求超时时间

-

1

+

f(x)

分钟

请求超时时间

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

数据消费模式

repeatable

i

绑定安全网关

无

配置说明

HTTP Request 操作配置包括基本配置、高级配置两项：

HTTP Request

配置备注

基本配置

请求URL

stringhttp://www.test.comf(x)

指定发送请求的URL

请求方法

GETf(x)

指定发送请求的方法

请求头

dict点击构造dictf(x)

指定发送请求的请求头,支持表达式、dict

请求参数

dict点击构造dictf(x)

指定发送请求的请求参数,支持表达式、dict,返回字典类型

最大重定向数

1f(x)

最大重定向数

请求超时时间

1f(x)分钟

请求超时时间

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

数据消费模式

repeatable

i

绑定安全网关

安全网关名称

无

基础配置参数描述

参数	参数类型	描述	是否必填	默认值
请求URL	string/表达式	用于指定发送请求的URL	是	/
请求方法	string	下拉单选指定发送请求的方法，全集为：GET、POST、PUT、PATCH、DELETE、HEAD	是	GET
请求头	表达式	用于指定发送请求的请求头，结构为：dict<string,list>	否	无
请求参数	表达式	用于指定发送请求的请求参数，结构为：dict<string,string>	否	无
最大重定向数	init/表达式	用于指定发送请求的请求体	否	1

请求超时时间	string/表达式	用于设置请求超时时间	否	1
--------	------------	------------	---	---

高级配置参数描述

参数	参数类型	描述	是否必填	默认值
消息属性	string	下拉框选择输出消息绑定的属性，HTTP Request 只支持绑定 payload	否	payload
类型选择	string	下拉框选择输出消息 payload 绑定的数据类型，包括 message 内置数据类型和自定义数据类型	否	空
数据消费模式	string	下拉框选择是否对消息数据进行持久化处理： repeatable：默认处理方式，消息数据可在一次触发中持续重复使用 non-repeatable：数据一次性消费，用于大文件数据的流式高效处理，暂时支持 JSON 数组类型，其超过5M时建议选择该模式	否	repeatable
安全网关名称	string	直接请求内网域名/IP 时，需要绑定安全网关	否	空

输出

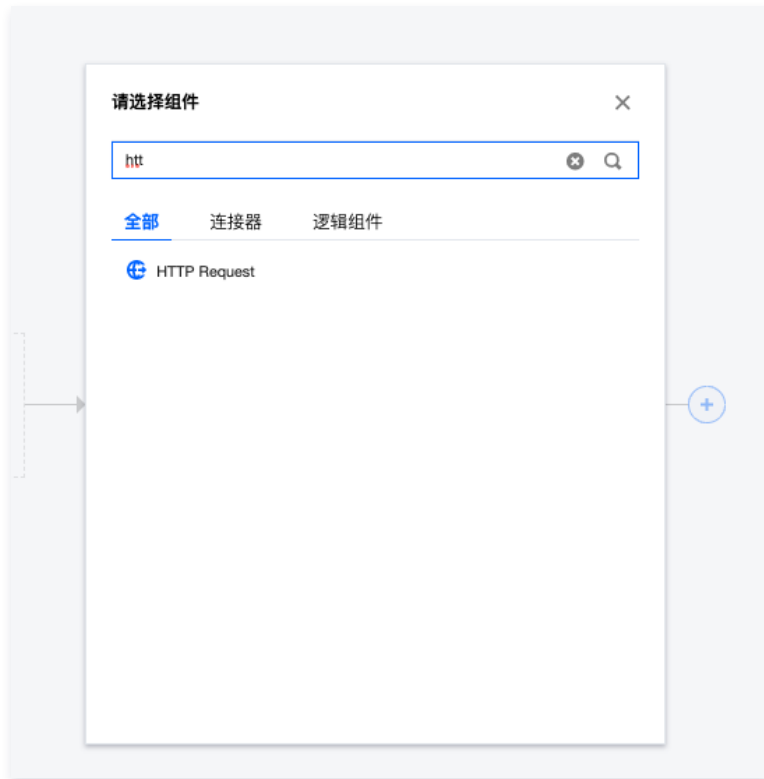
HTTP Request 请求的响应返回后，会将响应结果生成对应的消息传递给下一个组件。其中，响应的基本信息、Header、Query 参数、Path 参数会放到消息的 attributes 中，响应的 body 会放到消息的 payload 中。

HTTP Request 输出消息描述

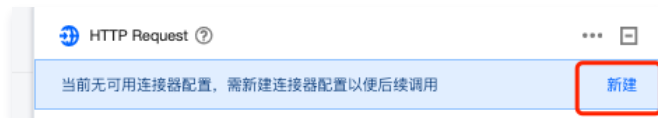
消息属性	值
payload	响应 body 保存在 payload 中，支持所有内置消息类型和自定义消息类型，其中 application/json、application/xml、application/yaml、application/x-www-form-urlencoded、multipart/form-data 等结构的 payload 支持在表达式中结构化访问
error	执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attributes	保存响应的属性值，具体包括： statusCode：响应的状态码，例如：200 reasonPhrase：响应的文本描述，例如：OK headers：响应的 header，结构为：dict<string,list> cookies：响应的 cookie，结构为：dict<string,string>，解析 Cookies 逻辑比较常见，可通过该数据直接读取
variables	默认为空

案例

1. 组件筛选 HTTP Request。



2. 新建并配置 HTTP Request 连接器，请求域名 `www.qq.com`。（此案例参考方法一进行配置）



配置界面如下：



3. 使用操作配置。

HTTP Request

...

当前使用的是 request-config 物理配置

编辑 | 切换

配置

调试

备注

基本配置

请求路径 *

string /

f(x)

指定发送请求的路径

请求方法 *

GET

f(x)

指定发送请求的方法

请求头

def dw_process(msg): return

🗑️

指定发送请求的请求头,支持表达式

请求参数

def dw_process(msg): return

🗑️

指定发送请求的请求参数, 只支持表达式, 返回字典类型

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

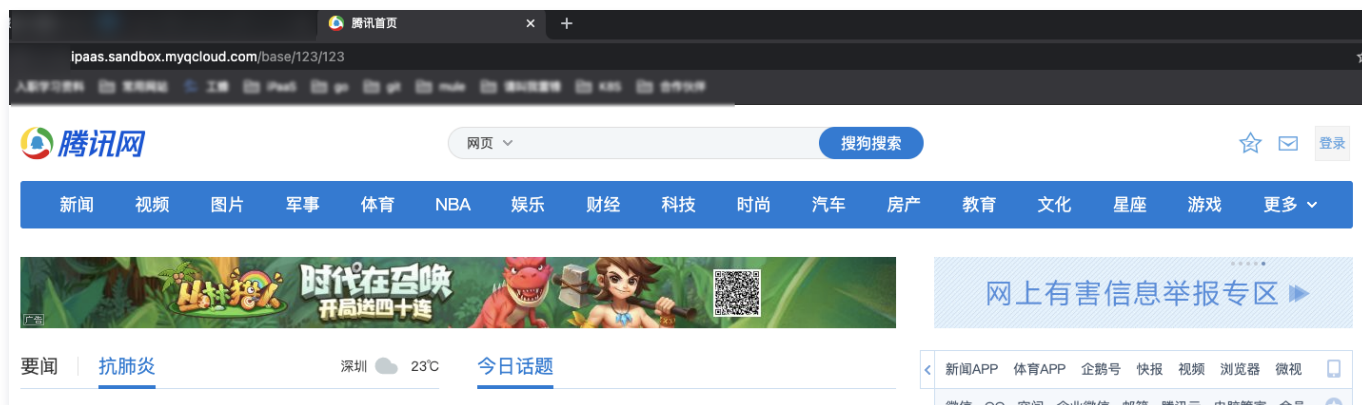
空

数据消费模式

repeatable

①

4. 发布并触发。浏览器访问触发对应流，可以直接跳转至腾讯首页。



同时，应用测试模式下可以看到 HTTP Request 对应的 attributes。

输出消息 attributes ▾

```
{
  "cookies": {},
  "headers": {
    "Cache-Control": [
      "max-age=60"
    ],
    "Content-Type": [
      "text/html; charset=GB2312"
    ],
    "Date": [
      "Mon, 26 Apr 2021 13:23:20 GMT"
    ],
    "Expires": [
      "Mon, 26 Apr 2021 13:24:20 GMT"
    ],
    "Server": [
      "squid/3.5.24"
    ],
    "Vary": [
      "Accept-Encoding",
      "Accept-Encoding",
      "Accept-Encoding"
    ],
    "X-Cache": [
      "HIT from shenzhen.qq.com"
    ]
  },
  "reasonPhrase": "200 OK",
  "statusCode": 200
}
```

方法三（2022年4月8日后创建的应用）

连接配置

- HTTP Request 连接器连接配置中包括 Base URL、证书验证、最大重定向次数及安全网关名称。

新建连接配置 ×

Base URL

请输入字符串

f(x)

证书验证 *

☒

最大重定向数

—

1

+

f(x)

请求超时时间

—

1

+

f(x)

分钟

安全网关名称

无

▾

完成

- 上述连接器配置参数相关描述如下表所示：

参数	数据类型	描述	是否必填	默认值
Base URL	string	一般配置请求地址的公共部分	否	无
证书验证	string	开启后则请求HTTP服务时跳过证书验证	是	HTTPS
最大重定向数	int	重定向次数，最小值1，最大值10	否	1
请求超时时间	string	设置请求的超时时间，最大5分钟，最小1分钟	否	1
安全网关名称	string	直接请求内网域名/IP 时，需要绑定安全网关	否	无

配置说明

HTTP Request ?

...

☐

当前使用的是 http:request-config#2 连接配置

查看 | 切换

配置

备注

基本配置

请求URL *

http://www.tenc.../getperson

f(x)

如果在连接配置中配置了BaseUrl,请配置相对URL.否则配置完整的请求URL

请求方法 *

POST

▼

f(x)

指定发送请求的方法

URL参数

dict

▼

点击构造dict

f(x)

指定发送请求的URL参数,支持表达式、dict,返回字典类型

请求头

dict

▼

点击构造dict

f(x)

指定发送请求的请求头,支持表达式、dict

请求体

string

▼

请输入字符串

f(x)

指定发送请求的请求体,支持字面量和表达式

高级配置

输出消息绑定自定义数据类型

消息属性

payload

▼

类型选择

空

▼

+

数据消费模式

repeatable

▼

ⓘ

- 基本配置：HTTP Request 连接器基本配置中包括请求URL、请求方法、URL参数、请求头、请求体，其中请求URL、请求方法为必填项。
- 高级配置：HTTP Request 连接器高级配置中输出消息绑定自定义数据类型，可用于自定义输出消息数据类型。

HTTP Request ?

...

新建

配置

备注

基本配置

请求URL *

string

http://www.xxx.com

f(x)

如果在连接配置中配置了Base URL,请配置相对URL.否则配置完整的请求URL

请求方法 *

GET

f(x)

指定发送请求的方法

URL参数

dict

点击构造dict

f(x)

指定发送请求的URL参数, 支持表达式、dict, 返回字典类型

请求头

dict

点击构造dict

f(x)

指定发送请求的请求头,支持表达式、dict

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

数据消费模式

repeatable

?

≡

基础配置参数描述				
参数	参数类型	描述	是否必填	默认值
请求URL	string/表达式	如果在连接配置中配置了BaseUrl, 请配置相对URL, 否则配置完整的请求URL	是	/
请求方法	string	下拉单选指定发送请求的方法, 全集为: GET、POST、PUT、PATCH、DELETE、HEAD、OPTIONS	是	GET
URL 参	表达式	用于指定发送请求的 URL 参数, 结构为: dict<string,string>	否	无

数				
请求头	表达式	用于指定发送请求的请求头，结构为：dict<string,list>	否	无
请求体	string/表达式	用于设置发送请求的消息体	否	无

高级配置参数描述

参数	参数类型	描述	是否必填	默认值
消息属性	string	下拉框选择输出消息绑定的属性，HTTP Request 只支持绑定 payload	否	payload
类型选择	string	下拉框选择输出消息 payload 绑定的数据类型，包括 message 内置数据类型和自定义数据类型	否	空
数据消费模式	string	下拉框选择是否对消息数据进行持久化处理： repeatable：默认处理方式，消息数据可在一次触发中持续重复使用 non-repeatable：数据一次性消费，用于大文件数据的流式高效处理，暂时支持 JSON 数组类型，其超过5M时建议选择该模式	否	repeatable

输出

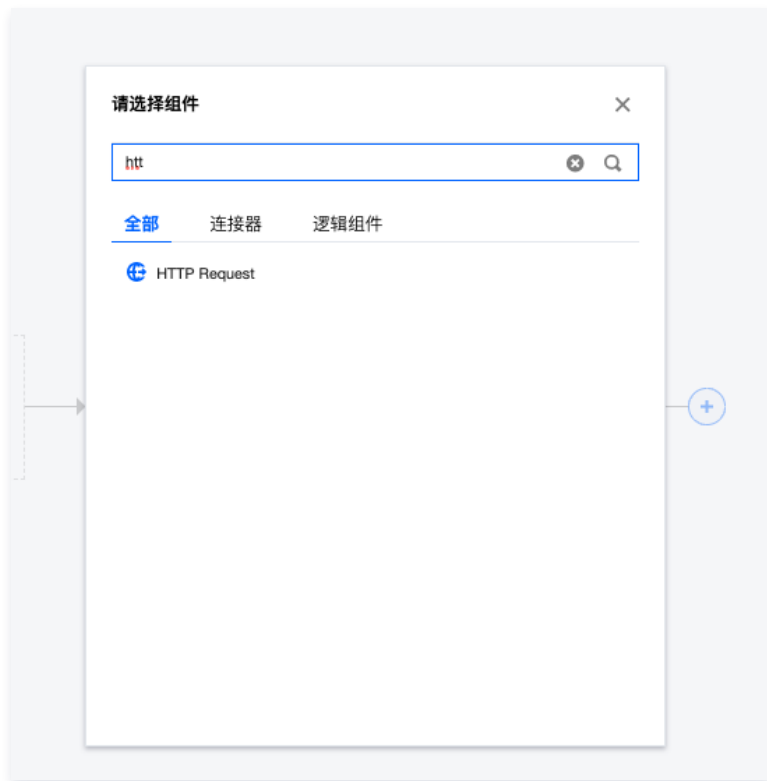
HTTP Request 请求的响应返回后，会将响应结果生成对应的消息传递给下一个组件。其中，响应的基本信息、Header、Query 参数、Path 参数会放到消息的 attributes 中，响应的 body 会放到消息的 payload 中。

HTTP Request 输出消息描述

消息属性	值
payload	响应 body 保存在 payload 中，支持所有内置消息类型和自定义消息类型，其中 application/json、application/xml、application/yaml、application/x-www-form-urlencoded、multipart/form-data 等结构的 payload 支持在表达式中结构化访问
error	执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attributes	保存响应的属性值，具体包括： statusCode：响应的状态码，例如：200 reasonPhrase：响应的文本描述，例如：OK headers：响应的 header，结构为：dict<string,list> cookies：响应的 cookie，结构为：dict<string,string>，解析 Cookies 逻辑比较常见，可通过该数据直接读取
variables	默认为空

案例

1. 新增组件时，筛选 HTTP Request。



2. 新建并配置 HTTP Request 连接器，BaseURL 配置为：`https://cloud.tencent.com`。



配置界面如下：

连接配置详情

×

基本信息

连接器

连接器版本

连接配置名称 http.request-config#3

连接配置分组 默认分组

配置详情

Base URL

https://cloud.tencent.com

f(x)

证书验证

最大重定向数

1

f(x)

请求超时时间

1

f(x)

 分钟

安全网关名称

无

编辑

取消

3. 使用操作配置。

HTTP Request ?

...

当前使用的是 http.request-config#3 连接配置

查看 | 切换

配置

备注

基本配置

请求URL

https://cloud.te...

/

f(x)

如果在连接配置中配置了BaseUrl,请配置相对URL.否则配置完整的请求URL

请求方法

GET

f(x)

指定发送请求的方法

URL参数

dict

点击构造dict

f(x)

指定发送请求的URL参数,支持表达式、dict,返回字典类型

请求头

dict

点击构造dict

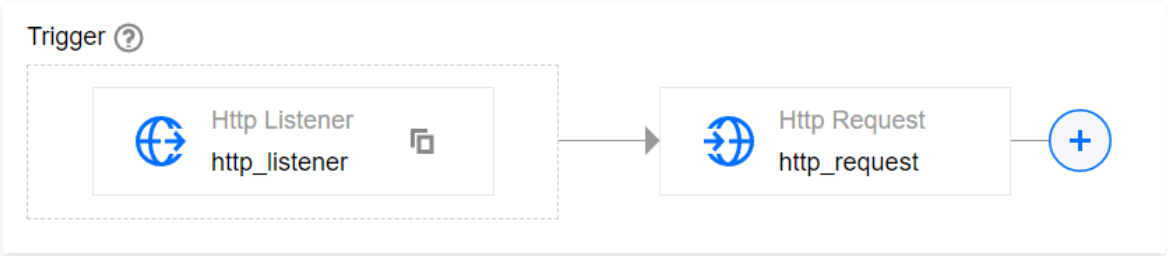
f(x)

指定发送请求的请求头,支持表达式、dict

4. 发布并触发。浏览器访问触发对应流，可以直接跳转至腾讯云首页。



集成流如下：



同时，应用测试模式下可以看到 HTTP Request 对应的 payload、attributes。

- payload：

HTTP Request ?

配置 调试 备注

运行消息

历史消息

2022-03-30 22:23:12.4

输入消息 payload 复制

""

下载

输出消息 payload 复制

"<!DOCTYPE html>\n<html lang=\"zh\">\n<head>\n<meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">\n<meta name=\"viewport\" content=\"width=device-width, maximum-scale=1\">\n<script>var timeStamps = [new Date()];</script>\n<meta charset=\"utf-8\">\n<title>腾讯云 - 产业智变 云启未来</title>\n<meta name=\"keywords\" content=\"腾讯云,云服务器,云主机,CDN,对象存储,域名注册备案,云存储,云数据库,互联网+解决方案,QQ云\">\n<meta name=\"description\" content=\"腾讯云为行业 and 开发者提供安全稳定的云计算服务,云服务器、云数据库、云存储、视频与CDN、域名注册\">\n</head>\n<body>\n</body>\n</html>\n"

下载

Mock ? ☐

以下变量值, 请至少配置1项

payload值

string

请输入字符串

f(x)

attributes:



SOAP 连接器

最近更新时间：2023-06-08 14:48:02

简介

- SOAP（简单对象访问协议：Simple Object Access Protocol）是基于 XML 的简易协议，可使应用程序在 HTTP 之上进行信息交换，用于访问网络服务。SOAP 提供一种标准的方法，使得运行在不同的操作系统并使用不同的技术和编程语言的应用程序可以互相进行通信。
- SOAP = HTTP + XML，本质即遵守 SOAP 编码规则的 HTTP 请求/响应。一条 SOAP 消息即为遵循 SOAP 协议规则的 XML 文档。主要包含下列元素：
 - 必需的 Envelope 元素，用于把此 XML 文档标识为一条 SOAP 消息。
 - 可选的 Header 元素，包含头部信息。如果 Header 元素被提供，则它必须是 Envelope 元素的第一个子元素。
 - 必需的 Body 元素，包含所有的调用和响应信息。
 - 可选的 Fault 元素，提供有关在处理此消息所发生错误的信息。
- WSDL 是基于 XML 的用于描述 Web Services 以及如何访问 Web Services 的语言。WSDL 是一种使用 XML 编写的文档，该文档可描述某个 Web service。它可规定服务的位置，以及此服务提供的操作（或方法）。
- SOAP 连接器即基于 SOAP 协议规则向 Web Services 发送请求的连接器。

连接器配置

通用配置				
参数	数据类型	描述	是否必填	默认值
WSDL Location	string	Web 服务的 WSDL URL 链接	是	–
WSDL 文件	string	Web 服务的 WSDL 文件	否	–
使用 WSDL URL	bool	是否使用 WSDL URL 链接进行分析	否	true
Service	string	Web 服务的 serviceName，通过 WSDL 文件获取	是	–
Port	string	Web 服务的 portName，通过 WSDL 文件获取	是	–
协议版本	enum	SOAP 协议版本：soap1.1、soap1.2	否	soap1.1
编码方式	string	SOAP Message 编码方式：UTF-8、UTF-16、ASCII、ISO-8859-1	否	UTF-8
请求超时时间（秒）	int	SOAP请求超时时间（秒）（范围：0~300）	否	60（s）

高级配置				
参数	数据类型	描述	是否必填	默认值
缓存过期时间（秒）	int	WSDL 文件缓存过期时间	否	90（s）

连接器配置界面如下：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

WSDL Location

string

请输入字符串

f(x)

WebService服务接口描述文件地址

使用WSDL URL

bool

true

Service

string

请输入字符串

f(x)

Port

string

请输入字符串

f(x)

SOAP配置

编码方式

UTF-8

f(x)

协议版本

soap11

f(x)

传输配置

请求超时时间(秒)

int

60

f(x)

高级

WSDL配置

wSDL缓存过期时间(秒)

int

90

f(x)

上一步

保存

操作配置

SOAP 请求操作如下：

输入参数

参数	数据类型	描述	是否必填	默认值
接口名称	string	请求 operation	是	—
请求体	string	SOAP Message body，只支持表达式输入	是	—
请求头	string	SOAP Message headers，只支持表达式输入	是	—
Web Service Security 使能	bool	是否通过可视化输入参数增加 Web Service Security 身份认证请求头	否	false
WsseMustUnderst and 属性	bool	请求接收者是否必须认可该请求头	否	true
UsernameTokenId 属性	string	Web Service Security 身份认证请求头 wsse:UsernameToken wsu:Id 属性值	否	—

版权所有：腾讯云计算（北京）有限责任公司

第420 共732页

用户名	string	Web Service Security身份认证请求头 wsse:Username 属性值	否	-
密码	string	Web Service Security身份认证请求头 wsse:Password 属性值	否	-
密码类型属性	enum	Web Service Security身份认证请求头 wsse:Password Type类型: PasswordText、PasswordDigest	否	Password Text
增加随机数属性	bool	是否使用自动生成的随机数作为 wsse:Nonce 属性值	否	false
用户自定义随机数	string	用户自定义输入 Web Service Security 身份认证请求头 wsse:Nonce 属性值, 此时增加随机数属性参数需设置为 false	否	-
增加创建时间属性	bool	是否使用自动生成的创建时间作为 wsu:Created 属性值	否	false
用户自定义创建时间	string	用户自定义输入 Web Service Security 身份认证请求头 wsu:Created 属性值, 此时增加创建时间属性参数需设置为 false	否	-

SOAP ？ SOAP请求

当前使用的是 soapbetterCfg 物理配置 编辑 | 切换

配置 备注

通用

接口名称

string 请输入字符串 f(x)

消息

请求体

请输入字符串 删除

请求头

请输入字符串 删除

高级

Web Service Security配置

Web Service Security使能

bool false

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空 +

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息

attribute	返回 SOAP 组件的 attribute 信息
variable	继承上个组件的 variable 信息

消息的 attributes:

attributes 信息	描述
attrs.statusCode	响应的状态码，例如：200
attrs.reasonPhrase	响应的文本描述，例如：OK
attrs.headers	响应的 Header，结果为 dict<string, list>
attrs.cookies	响应的 Cookie，结果为 dict<string, string>

消息的 Payload:

SOAP 请求返回的响应 Response 为 XML 格式，会首先进行一次 Flatten 处理，将 XML 转换为 dict 类型，然后放到消息 payload 中，后续可通过 dataway 表达式直接访问。

- 正确输出为 dict 类型，例如：

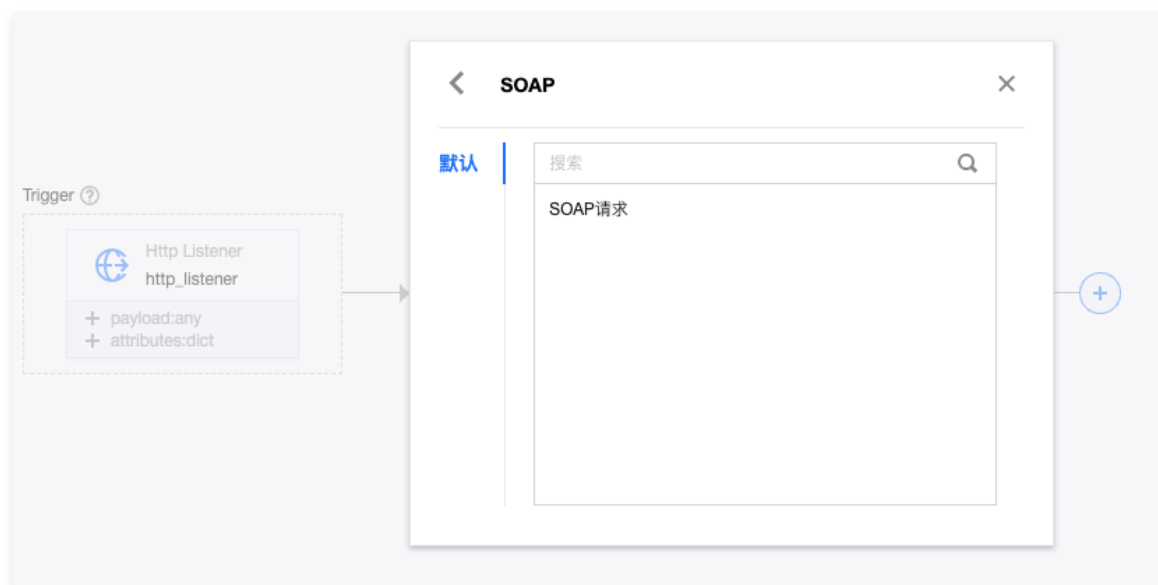
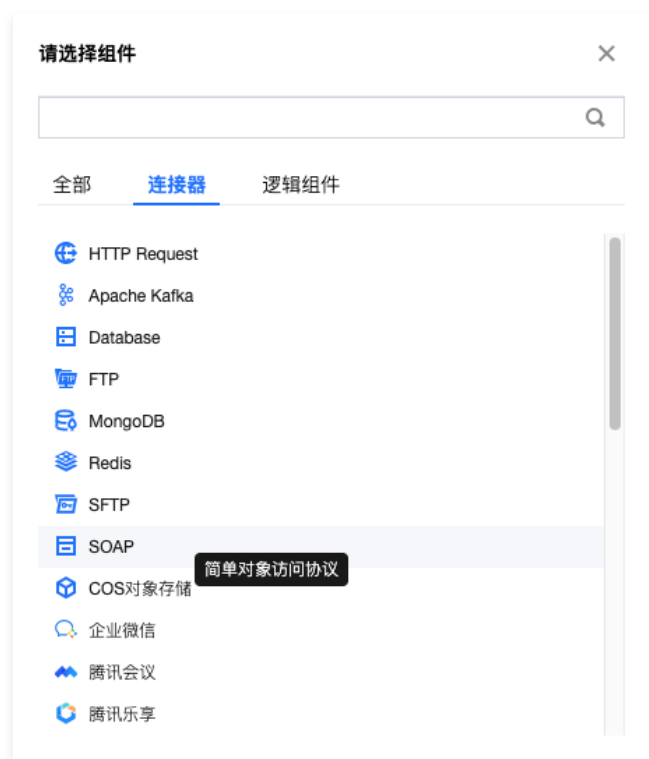
```
{
  "Body": {
    "AddResponse": {
      "AddResult": "5"
    }
  }
}
```

- 错误输出为 dict 类型，包含 “Code” 和 “Description” 元素，“Code” 表示错误类型，“Description” 表示错误具体信息，例如：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "SOAP:CONNECT Can not find method:Add1 in wsdl description"
}
```

案例

1. 在连接器列表中选择 SOAP 连接器，选择 SOAP 请求操作。



2. 输入连接器配置参数，其中 `WSDL Location = "http://www.dneonline.com/calculator.asmx?WSDL"` , `Service = "Calculator"` , `Port = "CalculatorSoap"` , 其他参

数保持默认值即可。具体如下图：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

WSDL Location

string

http://www.dneonline.com/calculat

f(x)

WebService服务接口描述文件地址

使用WSDL URL

bool

true

Service

string

Calculator

f(x)

Port

string

CalculatorSoap

f(x)

SOAP配置

编码方式

UTF-8

f(x)

协议版本

soap11

f(x)

传输配置

请求超时时间(秒)

int

60

f(x)

高级

WSDL配置

wsdl缓存过期时间(秒)

int

90

f(x)

上一步

保存

- 默认使用 WSDL URL 参数设置为 true，此时会根据 WSDL Location 参数值提供的 wsdl url 链接进行解析；当使用 WSDL URL 参数设置为 false 时，用户也可通过输入 WSDL 文件参数值来对 Web 服务的 WSDL 文件内容进行解析。

编辑连接器配置

✓

选择连接器

>

2

配置连接器

通用

连接配置

WSDL Location

string
http://www.dneonline.com/calculat
f(x)

WebService服务接口描述文件地址

WSDL文件

请输入字符串

WebService服务接口描述文件

使用WSDL URL

bool
false

Service

string
Calculator
f(x)

Port

string
CalculatorSoap
f(x)

SOAP配置

编码方式

UTF-8
f(x)

协议版本

soap11
f(x)

传输配置

请求超时时间(秒)

int
60
f(x)

高级

WSDL配置

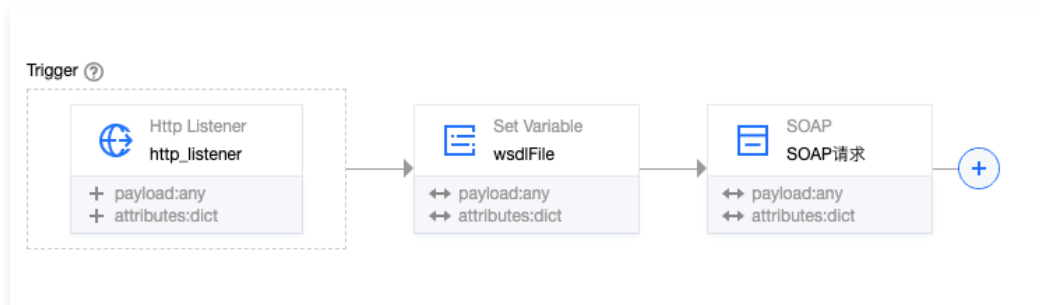
wSDL缓存过期时间(秒)

int
90
f(x)

上一步

保存

- 首先在 SOAP 组件前添加 SetVariable 组件，变量名设置为 wsdlFile，变量值类型选择 entity.xml、值设为 WSDL Url 链接的 xml 内容。



Set Variable ?

配置 备注

基本配置

变量名 * wsdlFile
变量的名称

变量值 * entity.xi This XML file does not appear to h f(x)

- 然后在 SOAP 组件连接器配置中将使用 WSDL URL 参数设置为 false，将 WSDL 文件参数设置为 `msg.vars["wsdlFile"]` 变量值保存即可。

编辑连接器配置

选择连接器 > 2 配置连接器

通用

连接配置

WSDL Location * string http://www.dneonline.com/calculat f(x)
WebService服务接口描述文件地址

WSDL文件 def dw_process(msg): return msg.vars["wsdlFi] f(x)
WebService服务接口描述文件

使用WSDL URL bool false

Service * string Calculator f(x)

Port * string CalculatorSoap f(x)

3. 确认连接器配置参数填写无误后单击保存，然后设置操作配置参数。接口名称设为“Add”（根据 WSDL 文件 operation 参数获取），请求头、请求体表达式输入见下图：

SOAP

SOAP请求

...

当前使用的是 soapworkCfg 物理配置

编辑

 |

切换

配置

备注

通用

▲

接口名称

string

Add

f(x)

消息

请求体

def dw_process(msg): res = { 'Add':{

请求头

def dw_process(msg): return "

高级配置

▲

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

○ SOAP 请求体表达式输入见下图：

Dataway表达式

1

2

3

4

def dw_process(msg):

res = [{ 'intA': 2 }, { 'intB': 3 }]

return res

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

attributes

attrs

- SOAP请求头表达式输入见下图，默认为空：

SOAP ? SOAP请求

当前使用的是 soapworkCfg 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

配置 / 编辑

Dataway表达式 ?

1 def dw_process(msg):

2 return

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

attributes

attrs

- 当需要支持 Web Service Security 身份认证方式时，SOAP 请求头表达式输入可参考下图进行配置：

Dataway表达式 ?

1 def dw_process(msg):

2 res = {

3 "wsse:Security": {

4 "@xmlns:wsse": "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd",

5 "@xmlns:wsu": "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd",

6 "@soap:mustUnderstand": "1",

7 "wsse:UsernameToken": {

8 "@wsu:Id": "UsernameToken-131234sdsdvsdvsdfbdgnfgjmdghm",

9 "wsse:Username": "ISU_JIRA_API@wwswcn2",

10 "wsse:Password": {

11 "@Type": "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText",

12 "#text": "*****"

13 },

14 "wsse:Nonce": {

15 "@EncodingType": "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary",

16 "#text": "riZ5dfdsqdhghjz12xmQ=="

17 },

18 "wsu:Created": "2021-02-19T12:13:31.598Z"

19 }},

20 }

21 }

22 return Entity.from_value(res, mime_type='application/xml')

自动联想

payload

payload

attributes

attrs

版权所有：腾讯云计算（北京）有限责任公司

第428 共732页

- 或者将 Web Service Security 使能设置为 true，进行 Web Service Security 身份认证请求头参数的可视化编辑，具体可参考下图进行配置：

SOAP ? SOAP请求

当前使用的是 soap11 物理配置

编辑 | 切换

配置

备注

通用

接口名称

string ▼ Add f(x)

消息

请求体

def dw_process(msg): res = [{"intA": 2}, {"intB": 2}]

请求头

def dw_process(msg): return

高级

Web Service Security配置

Web Service Security使能

bool ▼ true ▼

WsseMustUnderstand属性

bool ▼ true ▼

UsernameTokenId属性

UsernameToken-131234DB1D84EE2B71613wefhgsh:

用户名

ISU_JIRA_API@wwwswcn2

密码

密码类型属性

PasswordText ▼

增加随机数属性

bool ▼ false ▼

用户自定义随机数

riZ5ww9gXQsdkwdnzl2xmQ==

增加创建时间属性

bool ▼ false ▼

用户自定义创建时间

2021-02-19T12:13:31.598Z

4. 操作配置参数设置完成后保存返回即可，然后单击右上角发布，选择发布地域后单击确定。

于 1 分钟前保存

调试

发布

删除

请选择发布地域

天津

确定

取消

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 soapadd 物理配置 [查看](#)

配置 备注

基本配置

监听路径 * [复制监听域名](#)
指定监听的路径

监听方法 *
指定监听HTTP请求的方法

○ SOAP 返回 payload 响应结果如下，即为请求体中 intA 和 intB 两数相加结果。

← → ↺ soapadd-251196504.ipaas.beta.myqcloud.com/soap1

应用 彩云小译 - 在线翻译 Google 百度一下，你就知道 Career

`{"Body": {"AddResponse": {"AddResult": "5"}}`

输出消息 payload ▾

```
{
  "Body": {
    "AddResponse": {
      "AddResult": "5"
    }
  }
}
```

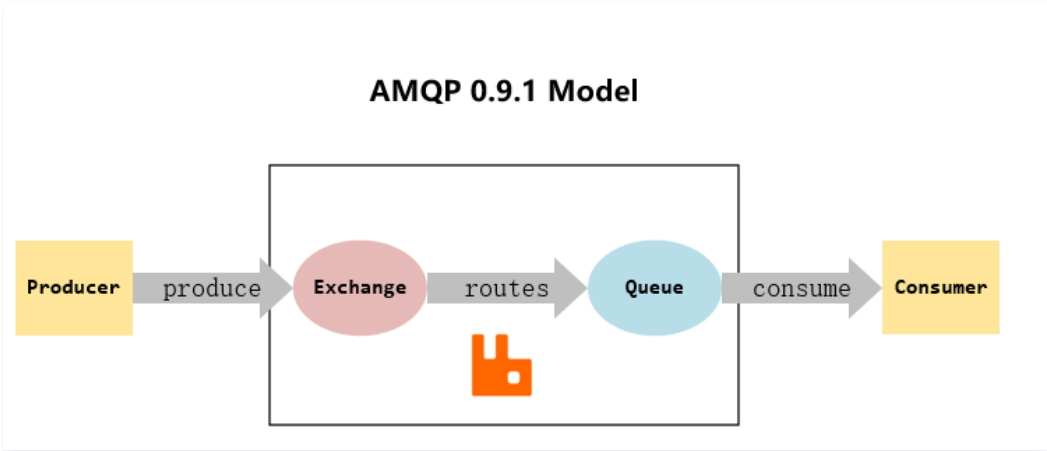
AMQP 连接器

最近更新时间：2023-06-08 14:48:02

简介

AMQP 连接器用于连接 RabbitMQ 集群和腾讯云数据连接器集群，可消费 RAbbitMQ 集群的消息并作为集成流的 trigger，也可将集成流的消息生产发布到 RabbitMQ 集群中。

AMQP 连接器遵从 AMQP 0.9.1 协议。AMQP 协议基本概念如下：



- **Producer**: 生产者，负责向 RabbitMQ 投递消息（实际是向交换机投递而非消息队列）。
- **RoutingKey**: 路由键，生产者投递消息时所指定的参数，用于交换机根据一定的路由规则转发消息到相关消息队列中。
- **Consumer**: 消费者，负责从 RabbitMQ 的消息队列中接收消息。
- **Exchange**: 交换机。生产者在向 RabbitMQ 投递消息时，并不能直接将消息投递到消息队列中。实际上，生产者是将消息投递到生产者所指定交换机上，不同类型的交换机具有其各自的路由规则，交换机根据一定的路由规则（使用消息所携带的 RoutingKey 和绑定在该交换机上的消息队列的 BindingKey），再将消息转发给相应的消息队列上。
- **Queue**: 消息队列，用于接收、储存交换机转发的消息，消费者通过其接收、消费消息。
- **BindingKey**: 绑定键，在消息队列绑定到交换机的过程中，可以指定 BindingKey 参数，用于交换机根据一定的路由规则转发消息到相关消息队列中。

连接器配置

连接配置参数				
参数	数据类型	描述	是否必填	默认值
集群地址	string	rabbitMQ 集群地址	是	-
集群端口	int	rabbitMQ 集群端口号	是	-
虚拟集群地址	string	虚拟集群地址	否	/
用户名	string	rabbitMQ 集群用户名	否	-
密码	string	rabbitMQ 集群密码	否	-
使能 TLS 安全传输协议	bool	是否使用 TLS 加密和 rabbitMQ 集群间的连接	否	false
TLS 客户端证书	file	可选，使用提供的证书对连接进行加密，仅当使 TLS 安全传输协议设置为 True 才可配置	否	-
TLS 客户端 Key	file	可选，使用提供的证书对连接进行加密，需和客户端证书同时提供，仅当使 TLS 安全传输协议设置为 True 才可配置	否	-
TLS 服务端证书	file	可选，使用提供的证书对连接进行加密，仅当使 TLS 安全传输协议设置为 True 才可配置	否	-

连接配置

集群地址 *

10.10.10.25

集群端口 *

int

5002

虚拟集群地址

/

安全认证模式

PLAIN

用户名

root

密码

.....

使能TLS安全传输协议

bool

true

f(x)

客户端证书文件内容

还未选择文件

选择文件

客户端key文件内容

还未选择文件

选择文件

服务端证书文件内容

还未选择文件

选择文件

通用配置

参数	数据类型	描述	是否必填	默认值
消息类型	enum	消息类型，默认为 text/plain	否	text/plain
消息编码格式	enum	消息编码格式	否	UTF-8

通用配置

消息类型

text/plain

▼

f(x)

消息编码格式

UTF-8

▼

f(x)

消费者配置

参数	数据类型	描述	是否必填	默认值
应答模式	enum	消息消费后的应答模式分为两种： 消费后直到触发的流成功结束后才确认 offset 消费后立即确认 offset	否	流运行成功后确认
消费模式	enum	消费模式，可选： 拉模式 推模式	否	拉模式
NoLocal	bool	设置为 true，表示不能将同一个 Connection 中生产者发送的消息传递给该 Connection 中的消费者	否	false
Exclusive	bool	设置为是否排他	否	false
消费者数量	int	设置消费者数量	否	1

消费者配置

应答模式

触发流之前确认

▼

消费模式

拉模式

▼

NoLocal

bool ▼

false

▼

Exclusive

bool ▼

false

▼

消费者数量

int ▼

1

生产者配置

参数	数据类型	描述	是否必填	默认值
投递模式	int	生产消息时的投递模式（持久化/不持久化）	否	持久化

生产者配置

投递模式

持久化

▼

f(x)

服务质量配置

参数	数据类型	描述	是否必填	默认值
预取大小（字节）	int	预取大小	否	0
预取数量	int	预取数量	否	0

服务质量配置

预取大小(字节)

int

▼

0

f(x)

预取数量

int

▼

0

f(x)

操作配置

AMQP 连接器包含 Consumer 和 Publish 两种操作。

Consumer 操作

输入参数

参数	数据类型	说明	是否必填	默认值
队列名称	string	消费的队列名称	是	-
消费者标识	string	消费者标识	否	-
NoWait	bool	为 true 时，client 端将不会等待 server 的 response，server 也不会 response；若 server 无法完成消费请求，则会抛出 channel 或 connection异常。	否	true

AMQP

AMQP Con...

...

当前使用的是 an 物理配置

编辑 | 切换

配置

备注

通用 ▲

队列名称 *

test

消费者标识

test

NoWait

bool

true

输出

Consumer 操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 byte 数组
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	执行成功后，attribute 为 dict，属性列表见下表
variable	继承上个组件的 variable 信息

atrrbites 属性列表：

atrrbites 属性	数据类型
headers	map[string]interface{}
deliveryMode	uint8, non-persistent (1) or persistent (2)
priority	uint8, 0 to 9
correlationId	string
replyTo	string
expiration	string
messageId	string
timestamp	time.Time
type	string
userId	string
appId	string

例如：执行成功后，message payload 值为 byte 数组，message attribute 值如下：

```
{
  "headers": {},
  "deliveryMode": 1,
  "priority": 4,
  "correlationId": "someId",
  "replyTo": "some_replyto",
  "expiration": "messageId",
  "timestamp": "2021-04-25T15:30:30Z08:00",
  "type": "",
  "userId": "",
  "appId": ""
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "some error message."
}
```

案例

1. 按照 rabbitMQ 集群信息，填写 AMQP 连接器配置的连接配置，其他配置保持默认。

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

连接配置

集群地址

9.14.0.25

集群端口

int

502

虚拟集群地址

/

安全认证模式

PLAIN

用户名

g

密码

使能TLS安全传输协议

bool

false

f(x)

通用配置

2. 新建流，将 AMQP Consumer 作为 trigger 节点，配置待消费的队列名称。

AMQP ? AMQP Con...

当前使用的是 amqp 物理配置 编辑 | 切换

配置 备注

通用

队列名称 *

logs

消费者标识

请输入字符串

NoWait

bool true

3. 添加其他业务逻辑，处理消费到的数据。例如：发布到 kafka。

配置中 20210426_1.0

Trigger ?

AMQP AMQP Consumer

payload:any attributes:dict

→

Apache Kafka Apache Kafka Producer

payload:any attributes:dict

+

Apache Kafka ? Apache Kaf...

当前使用的是 kafka 物理配置 编辑 | 切换

配置 备注

通用

通用配置

要写入的 Kafka主题名称

string logs fix

要写入的 kafka消息内容

string def dw_process(msg): return msg 消息数据类型只能是string或者byte数组

高级配置

目标partition

string 请输入字符串 fix

Key

string 请输入字符串 fix key相同的message会写入同一partition

header

string 请输入字符串 fix

高级配置

4. 若参数配置正确，流发布后，即可将 rabbitMQ 的消息消费并发布到 kafka 队列中。

Publish 操作

输入参数

通用配置

参数	数据类型	说明	是否必填
Exchange 名称	string	AMQP 模型的 Exchange 名称	是
Exchange 类型	enum	AMQP 模型的 Exchange 类型	是
路由键	string	生产者投递消息时所指定的参数，用于交换机根据一定的路由规则转发消息到相关消息队列中	否
上下文 ID	string	上下文 ID	否

AMQP ? AMQP Pro...

...

当前使用的是 amqp 物理配置

编辑 | 切换

配置备注

通用

Exchange名称 *

string ▼

info

f(x)

Exchange类型

topic ▼

f(x)

路由键

string ▼

route_key

f(x)

上下文ID

string ▼

unique_id

f(x)

Exchange 配置

参数	数据类型	说明	是否必填	默认值
不等待服务端确认投递结果（NoWait）	bool	设置为 True 时，客户端不等待服务端确认投递结果	否	false
是否持久化	bool	设置是否持久化	否	true
是否自动删除	bool	设置是否自动删除	否	false

AMQP ? AMQP Pro...

...

☐

当前使用的是 amqp 物理配置 [编辑](#) | [切换](#)

配置

备注

Exchange配置

不等待服务端确认投递结果

bool ▼ false ▼ f(x)

是否持久化

bool ▼ true ▼ f(x)

是否自动删除

bool ▼ false ▼ f(x)

是否为无法路由的消息进行返回处理

bool ▼ false ▼ f(x)

是否对路由到无消费者队列的消息进行返回处理

bool ▼ false ▼ f(x)

消息配置

参数	数据类型	说明	是否必填	默认值
优先级	int	设置消息优先级	否	0
消息标识 ID	string	设置消息标识 ID	否	-
应答队列名称	string	设置应答队列名称	否	-
消息存活时间（毫秒）	int	设置消息存活时间	否	1800000
生产者用户 ID	string	生产者用户 ID	否	-
生产者应用 ID	string	生产者应用 ID	否	-
消息类型	string	消息的 MIME 类型	否	-
消息头	map	消息头，字典类型，可添加用户自定义 header	否	-
消息体	string	消息体，投递的具体消息内容	否	-

AMQP

AMQP Pro...

...

[-]

当前使用的是 **amqp** 物理配置

编辑 | 切换

配置

备注

消息配置

优先级

int ▼

0

f(x)

消息标识ID

string ▼

请输入字符串

f(x)

应答队列名称

string ▼

请输入字符串

f(x)

消息存活时间
(毫秒)

int ▼

1800000

f(x)

生产者用户ID

string ▼

请输入字符串

f(x)

生产者应用ID

string ▼

请输入字符串

f(x)

消息响应时间

time ▼

16:07:03

🕒

f(x)

消息类型

string ▼

请输入字符串

f(x)

消息头

☒ 列表添加

☐ f(x)

value

操作

+添加

消息体

string ▼

test 123

f(x)

输出

Publish 操作执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "some error message."
}
```

案例

1. 按照 rabbitMQ 集群信息，填写 AMQP 连接器配置的连接配置，其他配置保持默认。

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

连接配置

集群地址 *

9.14.0.225

集群端口 *

int 5012

虚拟集群地址

/

安全认证模式

PLAIN

用户名

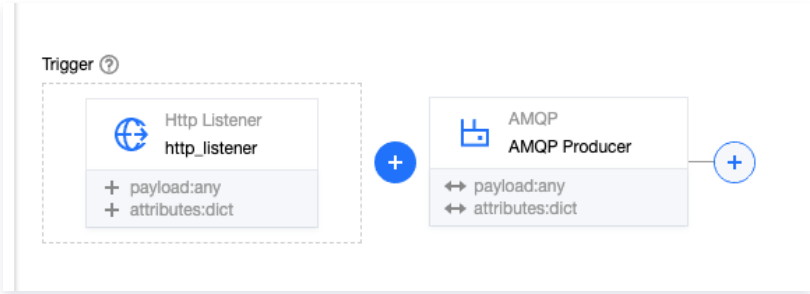
密码

使能TLS安全传输协议

bool false f(x)

通用配置

2. 新建流，添加 trigger，例如：http_listener。



3. 在流中增加 amqp publish 操作，填写消息属性及消息内容。

AMQP ? AMQP Pro...

当前使用的是 amqp 物理配置 编辑 | 切换

配置

备注

通用

Exchange名称

string

info

f(x)

Exchange类型

topic

f(x)

路由键

string

route_key

f(x)

上下文ID

string

unique_id

f(x)

AMQP ? AMQP Pro...

当前使用的是 amqp 物理配置 编辑 | 切换

配置

备注

是否为无法路由的消息进行返回处理

bool

false

f(x)

是否对路由到无消费者队列的消息进行返回处理

bool

false

f(x)

消息配置

优先级

int

0

f(x)

消息标识ID

string

请输入字符串

f(x)

应答队列名称

string

请输入字符串

f(x)

消息存活时间(毫秒)

int

1800000

f(x)

生产者用户ID

string

请输入字符串

f(x)

生产者应用ID

string

请输入字符串

f(x)

消息响应时间

time

16:07:03

f(x)

消息类型

string

请输入字符串

f(x)

消息头

列表添加

f(x)

value

操作

+添加

消息体

string

test 123

f(x)

4. 发布流，触发流后，若连接参数及消息参数配置正确，消息将成功投递到所配置的 rabbitMQ 集群的对应队列中。

MongoDB 连接器

最近更新时间：2023-06-08 14:48:02

简介

MongoDB 是 [面向文档](#) 的数据库管理系统，MongoDB 将数据存储为一个文档，数据结构由键值对（key/value）组成。MongoDB 文档类似于 JSON 对象，字段值可以包含其他文档，数组及文档数组。

MongoDB 连接器可连接第三方 MongoDB 数据库并执行操作。用户通过连接器配置来配置 MongoDB 的连接参数，配置成功后便可执行 MongoDB 操作。

连接器配置

- MongoDB 连接器支持两种连接形式：
- 自定义配置方式，用户根据连接器参数进行参数配置。
 - URI字符串方式，用户输入 MongoDB 连接 URI，例如："mongodb://localhost:27017"。

通用标签页-连接配置

参数	数据类型	描述	是否必填	默认值
连接方式	enum	MongoDB 连接模式，支持自定义配置方式和 URI 字符串方式	是	无
URI 字符串	string	URI 字符串方式参数，MongoDB 数据库连接 URI，例如："mongodb://localhost:27017"	是	无
服务器地址	string	自定义配置方式参数，MongoDB 地址，格式为 host:port，以逗号相隔	是	无
数据库	string	自定义配置方式参数，待连接数据库	是	无
用户名	string	自定义配置方式参数，用于验证的用户名	否	无
密码	string	自定义配置方式参数，用于验证的密码	否	无
认证方式	string	自定义配置方式参数，连接 MongoDB 数据的认证方式，支持 SCRAM-SHA-1、SCRAM-SHA-256、PLAIN	否	SCRAM-SHA-1
认证源	string	自定义配置方式参数，MongoDB 数据认证的授权来源	否	无
副本集名称	string	自定义配置方式参数，MongoDB 复制集名称	否	无

通用

连接方式

自定义配置方式

连接配置

服务器地址

127.0.0.1:27017,127.0.0.1:27018

格式为host:port,以逗号相隔

数据库

test

用户名

admin

密码

123456

认证方式

SCRAM-SHA-1

认证源

test

副本集名称

replicaSet1

高级标签页-连接配置

参数	数据类型	描述	是否必填	默认值
读写超时时间（毫秒）	int	自定义配置方式参数，读写网络超时时间，默认时间单位为毫秒	否	30000
连接超时时间（毫秒）	int	自定义配置方式参数，连接 MongoDB 数据库的超时时间，默认时间单位为毫秒	否	30000
启用可重试写入	bool	自定义配置方式参数，是否可重试写入，默认为 false，不启用可重试写入	否	false
连接池最小连接数	int	自定义配置方式参数，连接池最小连接数	否	0
连接池最大连接数	int	自定义配置方式参数，连接池最大连接数	否	100
连接最大空闲时间（毫秒）	int	自定义配置方式参数，连接最大空闲时间，默认时间单位为毫秒	否	0

高级

连接配置

读写超时时间
(毫秒)

int

30000

连接超时时间
(毫秒)

int

30000

启用可重试写入

bool

false

连接池最小连接数

int

0

连接池最大连接数

int

100

连接最大空闲时间 (毫秒)

int

0

高级标签页-服务路由配置

参数	数据类型	描述	是否必填	默认值
服务器选择RTT参考阈值 (毫秒)	int	自定义配置方式参数, MongoDB 数据库服务器选择 RTT 参考阈值, 默认时间单位为毫秒	否	15
服务器选择超时时间 (毫秒)	int	自定义配置方式参数, MongoDB 数据库服务器选择超时时间, 默认时间单位为毫秒	否	30000

服务路由配置

服务器选择RTT参考阈值 (毫秒)

int

15

服务器选择超时时间 (毫秒)

int

30000

高级标签页-读策略

参数	数据类型	描述	是否必填	默认值
Read Concern	enum	自定义配置方式参数, 读关注类型设置, 支持 LOCAL、AVAILABLE、MAJORITY、LINEARIZABLE、SNAPSHOT	否	MAJORITY
Read Preference	enum	自定义配置方式参数, 读偏好类型设置, 支持 PRIMARY、PRIMARYPREFERRED、SECONDARY、SECONDARYPREFERRED、NEAREST	否	PRIMARY

e				
---	--	--	--	--

读策略

Read Concern

MAJORITY

Read Preference

PRIMARY

高级标签页-写策略

参数	数据类型	描述	是否必填	默认值
Write Concern	enum	自定义配置方式参数，写操作确认等级，对应 MongoDB 数据库 writeConcern 参数	否	majority
写确认超时时间（毫秒）	int	自定义配置方式参数，写操作超时时间，默认时间单位为毫秒	否	1000

写策略

Write Concern

majority

writeConcern w参数

写确认超时时间（毫秒）

int

1000

操作说明

MongoDB 连接器的操作分为集合操作、文档操作、索引操作。

← MongoDB

×

集合操作

文档操作

索引操作

搜索

判断集合存在

创建集合

删除集合

列出数据库所有集合

集合操作

判断集合存在

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 bool 类型，true 表示集合存在，false 表示集合不存在；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 MongoDB 连接器组件。



2. 新建连接器配置，填写配置参数，单击测试连接，测试连接器配置是否正确。

✓ 选择连接器

>

2 配置连接器

通用

连接方式

自定义配置方式

连接配置

服务器地址

3

格式为host:port,以逗号相隔

数据库

用户名

密码

认证方式

SCRAM-SHA-1

认证源

副本集名称

高级

安全网关

安全网关

无

3. 在通用配置中，填入集合名称。

MongoDB 判断集合存在

当前使用的是 mmm 物理配置

编辑 | 切换

配置备注

通用

集合名称 *

string

test-coll

f(x)

4. 执行成功后，message payload 中包含查询结果。

输出消息 payload ▾

false

创建集合

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
集合最大文档数	int	集合最大文档数	否	无
集合最大容量	int	集合最大容量，单位为 Byte	否	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 bool 类型，true 表示集合创建成功；若集合已存在或执行失败后，则 payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 MongoDB 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填入待删除的键。

MongoDB 创建集合

当前使用的是 mmm 物理配置

编辑 | 切换

配置

备注

通用

集合名称 *

string

test-coll

f(x)

集合最大文档数

int

请输入整数

集合最大容量
(单位:BYTE)

int

请输入整数

4. 执行成功后，message payload 中保存查询结果；执行失败后，则 Message error 保存报错信息。

• 执行成功：

输出消息 payload

true

• 集合已存在，则会报错：

输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "(NamespaceExists) Collection already exists. NS: test.test-coll"
}
```

删除集合

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 值为 bool 类型，true 表示集合删除成功；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 MongoDB 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。
4. 执行成功后，message payload 中保存执行结果；执行失败后，Message error 保存报错信息。

列出数据库所有集合

参数配置

不需要填写输入参数。

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。操作执行成功，返回保存集合名称的 list 数组信息。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 list 类型，成员为 dict 类型，保存集合信息；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：操作执行成功，message payload 如下：

```
[
  "test_coll2",
  "test_coll1",
  "test_coll3"
]
```

案例

1. 添加 MongoDB 连接器组件。
2. 新建连接器或选择已创建的连接器。

3. 执行成功后，message payload 中保存执行结果；执行失败后，Message error 则保存报错信息。上例执行成功，返回 list 数组信息：

输出消息 payload ▾

```
[
  "test_coll2",
  "test_coll",
  "test_coll3"
]
```

文档操作

插入文档

插入文档，支持批量插入文档。

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
文档信息	list	文档信息，list 类型，元素为 JSON 字符串格式	是	无
是否按集合顺序插入	bool	是否按集合顺序插入	否	false

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 list 类型，存储文档的 ObjectID 信息；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：文档插入成功，message payload 如下：

```
[
  "607e6f6f25e3aa9635c55ef2",
  "607e6f6f25e3aa9635c55ef3"
]
```

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

MongoDB ? 插入文档

当前使用的是 mongo-connector 物理配置
编辑 | 切换

配置	备注
通用	
集合名称 *	string test_coll f(x)
文档信息 *	☒ 列表添加 ☐ f(x)
value	操作
item_1	删除
item_2	删除
+添加	
是否按集合顺序插入	bool false

○ item1:

item_1

文档信息 *

string
{"name":"aaa","score":10}
f(x)

确定 取消

○ item2:

item_2

文档信息 *

string
{'name':'bbb','score':20}
f(x)

确定 取消

4. 执行成功后，message payload 中保存执行结果；执行失败后，Message error 保存报错信息。上例执行成功，返回 list。

输出消息 payload

```
[
  "6062d409b1840e0fe62e3e94",
  "6062d409b1840e0fe62e3e95"
]
```

查询文档

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
查询条件表达式	string	MongoDB 表达式格式，例如：{'field1': 'value1'}, {'field2': {'\$gte': 1, '\$lt': 10 }}	是	无
排序字段	string	排序字段	否	无
降序显示文档	bool	是否按排序字段降序显示文档	否	false
每页文档数量	int	查询返回的每页文档数量	否	无
最大文档数量	int	查询返回的最大文档数量	否	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 list 类型，成员为 dict 类型，保存文档的信息，键为字段名称，值为字段值，"_id" 字段为文档的 ObjectID 信息；若没有满足条件的文档或执行失败后，则 payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：文档查询成功，message payload 如下：

```
[
  {
    "_id": "607e63275e1e4e04f692d306",
    "name": "ccc",
    "score": 100
  },
  {
    "_id": "607e6f6f25e3aa9635c55ef2",
    "name": "aaa",
    "score": 10
  }
]
```

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

MongoDB 查询文档

当前使用的是 mongo-connector 物理配置

编辑 | 切换

配置

调试

备注

通用

集合名称

string

test_coll

f(x)

查询条件表达式

string

{'score':{'\$gte':2}}

f(x)

排序字段

string

score

降序显示文档

bool

true

每页文档数量

int

100

最大文档数量

int

100

4. 执行成功后，message payload 中保存执行结果；执行失败后，Message error 保存报错信息。上例执行成功，返回文档信息。

输出消息 payload

```
[
  {
    "_id": "6062d7efb1840e0fe62e3e9a",
    "name": "bbb",
    "score": 20
  },
  {
    "_id": "6062d7efb1840e0fe62e3e99",
    "name": "aaa",
    "score": 10
  }
]
```

获取满足指定条件的文档数

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
查询条件表达式	string	MongoDB 表达式格式，例如：{'field1': 'value1'}, {'field2': {'\$gte': 1, '\$lt': 10 }}	是	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	执行成功后，payload 值为 int 类型，表示满足查询条件的文档数量；若没有满足条件的文档或执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 MongoDB 连接器组件。
2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：

MongoDB 获取满足指...

当前使用的是 mongo-connector 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

集合名称

string

test_coll

f(x)

查询条件表达式

string

{"score":{"\$gte":2}}

f(x)

4. 执行成功后，message payload 中保存执行结果；执行失败后，Message error 保存报错信息。上例执行成功，返回文档数量。

输出消息 payload

2

更新文档

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
查询条件表达式	string	MongoDB 表达式格式，例如：{'field1': 'value1'}, {'field2': {'\$gte': 1, '\$lt': 10 }}	是	无
待更新内容	string	待更新文档信息，JSON 字符串格式	是	无
只更新第一个满足条件的文档	bool	是否只更新第一个满足条件的文档	否	无
没有满足查询条件的文档时，创建一个新文档	bool	没有满足查询条件的文档时，创建一个新文档	否	无

输出

操作执行成功后，输出结果会保存在Message消息体的payload；执行失败后，错误信息会保存在Message消息体的error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 dict 类型，包含以下字段：满足条件的文档数量 matchedCount、更新的文档数量 modifiedCount、新创建的文档数量 upsertedCount；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：操作执行成功，message payload 如下：

```
{
  "matchedCount": 3,
  "modifiedCount": 3,
  "upsertedCount": 0
}
```

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

MongoDB 更新文档

当前使用的是 mongo-connector 物理配置

配置

调试

备注

通用

集合名称

string

test_coll

f(x)

查询条件表达式

string

{'score':{'\$gte':2}}

f(x)

待更新内容

string

{'\$set':{'name':'jason'}}

f(x)

只更新第一个满足条件的文档

bool

true

没有满足查询条件的文档时，创建一个新文档

bool

false

4. 执行成功后，message payload 中保存查询结果；执行失败后，Message error 保存报错信息。上例执行成功，返回文档更新信息。

输出消息 payload ▾

```
{
  "matchedCount": 2,
  "modifiedCount": 2,
  "upsertedCount": 0
}
```

删除文档

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
查询条件表达式	int	MongoDB 表达式格式，例如：{'field1': 'value1'}, {'field2': {'\$gte': 1, '\$lt': 10 }}	是	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 int 类型，表示已删除的文档数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 添加 MongoDB 连接器组件。
- 新建连接器或选择已创建的连接器。
- 在通用配置中，填写相关参数。例如：

MongoDB ? 删除文档 ...

当前使用的是 mongo-connector 物理配置 编辑 | 切换

配置 调试 备注

通用 ▲

集合名称 *

string ▾

test_coll

f(x)

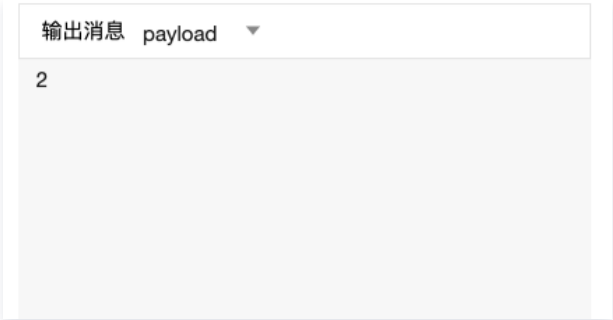
查询条件表达式 *

string ▾

{'score':{'\$gte':2}}

f(x)

4. 执行成功后，message payload 中保存查询结果；执行失败后，Message error 保存报错信息。上例执行成功，返回已删除的文档数量。



索引操作

创建单列索引

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
字段名称	string	字段名称	是	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为创建的索引名称；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。

3. 在通用配置中，填写相关参数。例如：

MongoDB ? 创建单列索引

当前使用的是 mongo-connector 物理配置 编辑 | 切换

配置 调试 备注

通用

集合名称 *

string test_coll f(x)

字段名称 *

string score

4. 执行成功后，message payload 中保存查询结果；执行失败后，Message error 保存报错信息。上例执行成功，索引名称：

输出消息 payload

"score_1"

查看索引

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 list 类型，成员为 dict 类型，保存索引信息；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：操作执行成功，message payload 如下：

```
[
  {
    "key": {
      "_id": 1
    },
  },
]
```

```
    "name": "_id_",
    "v": 2
  },
  {
    "key": {
      "score": 1
    },
    "name": "score_1",
    "v": 2
  }
]
```

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

MongoDB 查看索引

当前使用的是 mongo-connector 物理配置 编辑 | 切换

配置 调试 备注

通用

集合名称 * string test_coll f(x)

- 4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功，返回索引信息：

输出消息 payload

```
[
  {
    "key": {
      "_id": 1
    },
    "name": "_id_",
    "v": 2
  },
  {
    "key": {
      "score": 1
    },
    "name": "score_1",
    "v": 2
  }
]
```

下载

删除索引

参数配置

参数	数据类型	描述	是否必填	默认值
集合名称	string	集合名称	是	无
索引名称	string	索引名称	是	无

输出

操作执行成功后，输出结果会保存在 Message 消息体的 payload；执行失败后，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 值为 bool 类型，true 表示索引删除成功；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

- 1. 添加 MongoDB 连接器组件。
- 2. 新建连接器或选择已创建的连接器。
- 3. 在通用配置中，填写相关参数。例如：

MongoDB 删除索引

当前使用的是 mongo-connector 物理配置

配置 调试 备注

通用

集合名称 *

string

test_coll

f(x)

索引名称 *

string

score_1

- 4. 执行成功后，message payload 中保存查询结果；执行失败后，message error 保存报错信息。上例执行成功：

输出消息 payload

true

FTP 连接器

最近更新时间：2023-06-08 14:48:02

简介

- FTP（File Transfer Protocol，文件传输协议）是 TCP/IP 协议组中的协议之一。FTP 协议包括两个组成部分：FTP 服务器和 FTP 客户端。其中 FTP 服务器用来存储文件，用户可以使用 FTP 客户端通过 FTP 协议访问位于 FTP 服务器上的资源。在开发网站的时候，通常利用 FTP 协议把网页或程序传到 Web 服务器上。此外，由于 FTP 传输效率非常高，在网络上传输大的文件时，一般也采用该协议。
- 默认情况下，FTP 协议使用 TCP 端口中的20和21这两个端口，其中20用于传输数据，21用于传输控制信息（命令）。是否使用20作为传输数据的端口与 FTP 使用的传输模式有关，如果采用主动模式，则数据传输端口为20；如果采用被动模式，则最终使用的端口由服务器端和客户端协商决定。
- FTP 连接器是基于 FTP 协议实现文件传输的通道，因此应用程序才可与外部 FTP 服务器进行文件的交换。
- 针对 FTP 连接器提供对 FTP 服务器上的文件和文件夹的访问，主要特性包括：
 - 根据需要读写文件或完全列出目录内容的功能。
 - 支持常见的 FTP 操作，例如：创建目录和复制、移动、重命名和删除文件。
 - 支持锁定文件。
 - 支持文件匹配功能。

连接器配置

通用配置

参数	数据类型	描述	是否必填	默认值
域名	string	FTP 服务器域名	是	-
端口号	int	FTP 服务器端口号	否	21
用户名	string	FTP 服务器的用户名，如果服务器需要身份验证，则为必需参数	否	-
密码	string	FTP 服务器的密码，如果服务器需要身份验证，则为必需参数	否	-
工作路径	string	指定 FTP 服务器路径作为根目录。若未提供，则为 FTP 服务器默认值	否	-
传输模式	enum	传输模式，目前支持 BINARY（默认值）和 ASCII	否	BINARY
远程 Host 连接验证	bool	启用或禁用验证，判断进行数据连接的远程 host 与控制命令连接的 host 是否相同	否	True

高级配置

参数	数据类型	描述	是否必填	默认值
重连接频率（秒）	int	重连接频率（单位s），范围：0~300	否	10(s)
控制编码	enum	服务器通道控制编码，ISO-8859-1（默认）和 UTF-8	否	ISO-8859-1
连接超时时间（秒）	int	连接超时等待时间（单位s），范围：0~300	否	60(s)
响应超时时间（秒）	int	请求响应超时等待时间（单位s），范围：0~300	否	60(s)

连接器配置界面如下：

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

通用

连接配置

域名

请输入字符串

端口号

int

21

用户名

请输入字符串

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

操作配置

读取文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件路径	string	读取文件路径	是	
文件锁	bool	是否开启文件锁	否	false
文件锁等待时间（秒）	int	文件锁等待时间（秒），范围：0~60	否	30(s)
文件校验时间间隔（秒）	int	文件大小校验等待时间（秒），范围：0~60，以判断文件是否可读。在等待时间前后执行两次 sizecheck 文件大小检测，如果两次检测都返回相同的值，即 sizecheck 期间文件没有其他修改操作则可以读取该文件	否	0(s)

FTP 读取文件

当前使用的是 ftptestpreCfg 物理配置

配置 备注

通用

文件路径

string 请输入字符串 f(x)

高级

文件锁

bool false

文件锁等待时间(秒)

int 30

文件校验时间间隔(秒)

int 0

高级配置

输出消息绑定自定义数据类型

消息属性 payload

类型选择 空

输出

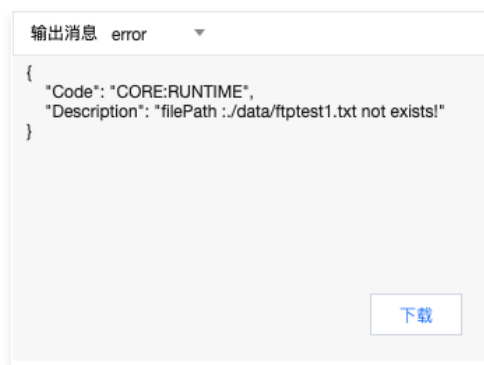
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 Entity 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

- FTP 读取文件操作会返回文件内容，根据文件名后缀设置消息的 mimeType 和 encoding 并放到消息 payload 中，后续可通过 dataway 表达式直接访问。



- FTP 读取文件操作若执行失败时输出消息 error 中会设置详细报错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：



案例

- 在连接器列表中选择 FTP 连接器，然后选择 FTP 读取文件操作。





2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

通用

连接配置

域名

007.3vftp.com

端口号

int

21

用户名

victorwsw

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为 `/data/ftpctest.txt`，参数输入见下图：

FTP 读取文件

当前使用的是 `ftpctestpreCfg` 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

通用

文件路径 `string` `/data/ftpctest.txt` `f(x)`

高级

文件锁 `bool` `true`

文件锁等待时间 (秒) `int` `30`

文件校验时间间隔 (秒) `int` `0`

高级配置

输出消息绑定自定义数据类型

消息属性 `payload`

类型选择 `空` `+`

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存 [调试](#) [发布](#) [删除](#) [保存](#)

请选择发布地域

☒ 天津

[确定](#) [取消](#)

5. 待集成流发布成功后复制HTTP Listener监听路径后访问该域名即可触发流。

Http Listener

当前使用的是 `ftp1` 物理配置 [查看](#)

配置 备注

基本配置

监听路径 `/ftp1` [复制](#)

指定监听的路径

监听方法 `全部`

指定监听HTTP请求的方法

复制监听域名

- FTP 返回 payload 响应结果如下，即为 “/data/ftpctest.txt” 文件内容。

← → ↻ ⚠ 不安全 | ftp1-251196504.ipaas.beta.myqcloud.com/ftp1

应用 彩云小译 - 在线翻译 Google 百度一下，你就知道 Career

Hello world!

输出消息 payload ▾

"Hello world!"

下载

- 当操作执行失败时，FTP 输出消息 error 中会设置详细报错信息。



500

CORE:RUNTIME

filePath ../data/ftpctest1.txt not exists!(Traceld:478938f5-eef5-86e8-fb2a-5e437f36e6c2)

输出消息 error ▾

```
{  "Code": "CORE:RUNTIME",  "Description": "filePath ../data/ftpctest1.txt not exists!"}
```

下载

写文件

输入参数

参数	数据类型	描述	是否必填	默认值
----	------	----	------	-----

文件路径	string	待写入文件路径（当检测为目录时抛错）支持表达式输入	是	
文件内容	string	待写入文件的内容，默认为当前消息 payload 支持表达式输入	否	#[payload]
创建父级目录	bool	是否尝试创建任何不存在的父目录	否	true
文件锁	bool	是否锁定文件	否	false
文件锁等待时间（秒）	int	文件锁等待时间（秒）（范围：0~60）	否	30(s)
文件写模式	enum: OVERWRITE、 APPEND、 CREATE_NEW	文件写模式： - OVERWRITE：覆盖写 - APPEND：追加写 - CREATE_NEW：当文件不存在时新建，否则抛错	否	OVERWRITE

FTP 写文件

当前使用的是 ftptestpreCfg 物理配置

配置

调试

备注

通用

文件路径

string

请输入字符串

f(x)

文件内容

string

请输入字符串

f(x)

创建父级目录

bool

true

文件锁

bool

false

文件锁等待时间 (秒)

int

30

文件写模式

OVERWRITE

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

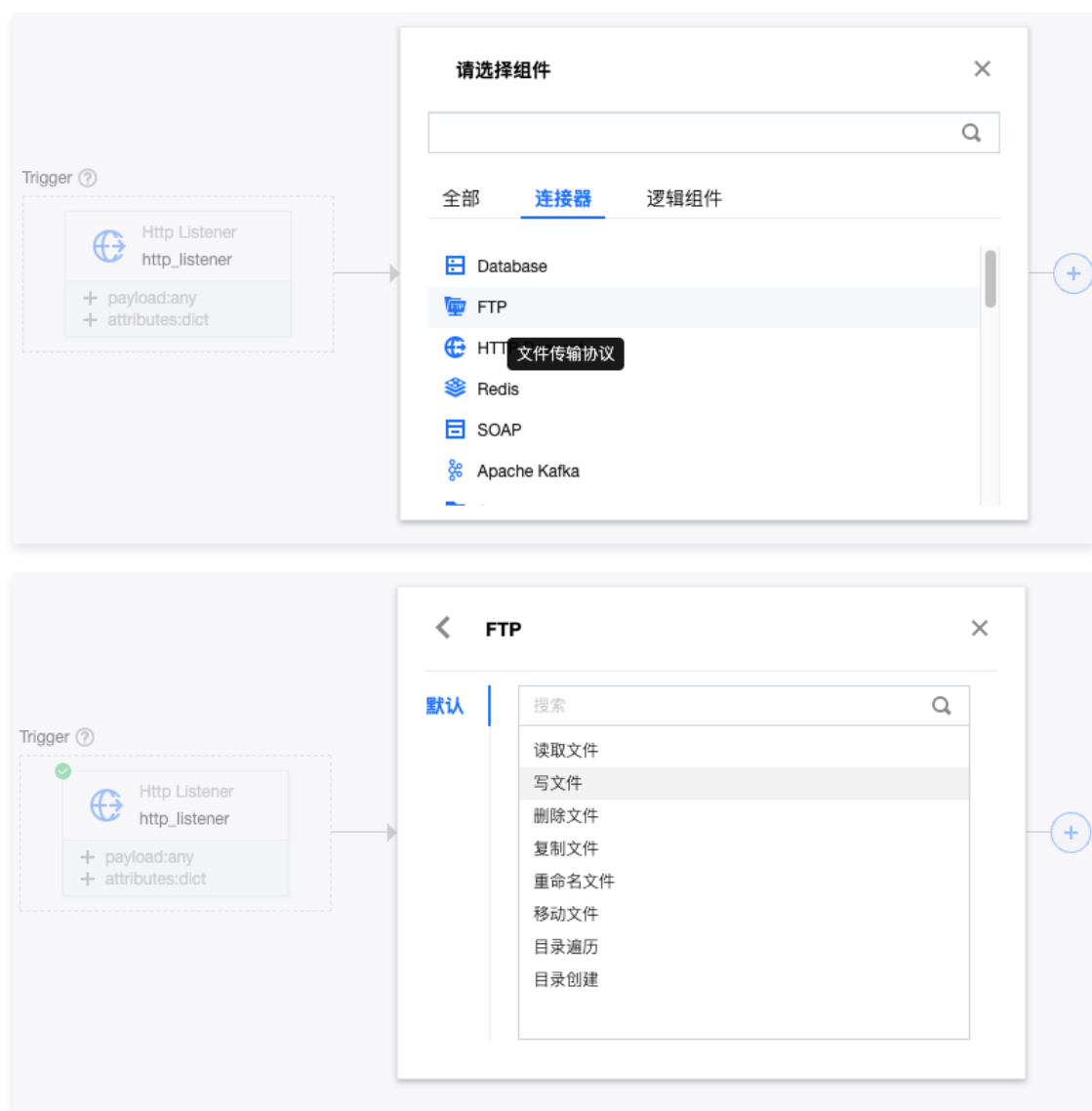
message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 写文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细报错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：



案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 写文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置



✓ 选择连接器 > 2 配置连接器

通用

连接配置

域名 *

端口号 int ▼

用户名

密码

通用配置

工作路径

传输模式 BINARY ▼

远程Host连接验证 bool ▼ true ▼

高级

重连接配置

重连接频率(秒) int ▼

通用配置

控制编码 ISO-8859-1 ▼

超时配置

连接超时时间(秒) int ▼

响应超时时间(秒) int ▼

测试连接

上一步

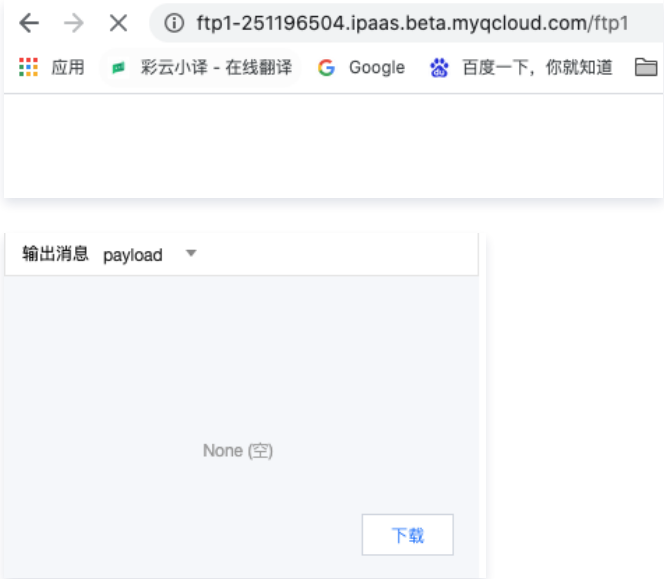
保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/ftpctest.txt”，参数输入见下图：

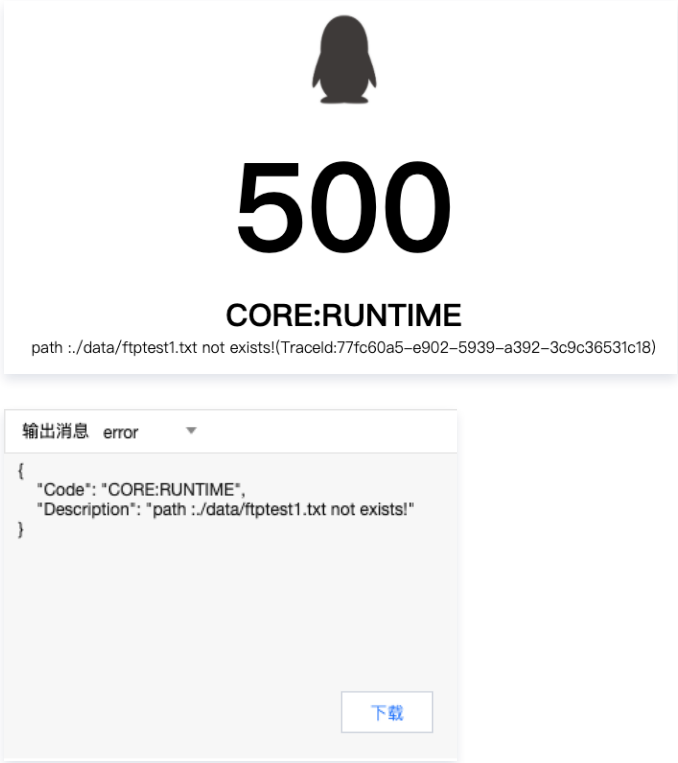
4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

- FTP 写文件操作若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，FTP 输出消息 error 中会设置详细返错信息。



删除文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件删除路径	string	待删除的文件或路径： - 当文件或路径不存在时抛错 - 支持删除单个文件以及目录（包括目录下所有文件） - 如果文件未被锁定，则删除路径指向的文件，支持表达式输入	是	-

FTP 删除文件

...

当前使用的是 `ftpctestpreCfg` 物理配置 [编辑](#) | [切换](#)

配置

调试

备注

通用

文件删除路径

string 请输入字符串

`f(x)`

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

 +

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 删除文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细报错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

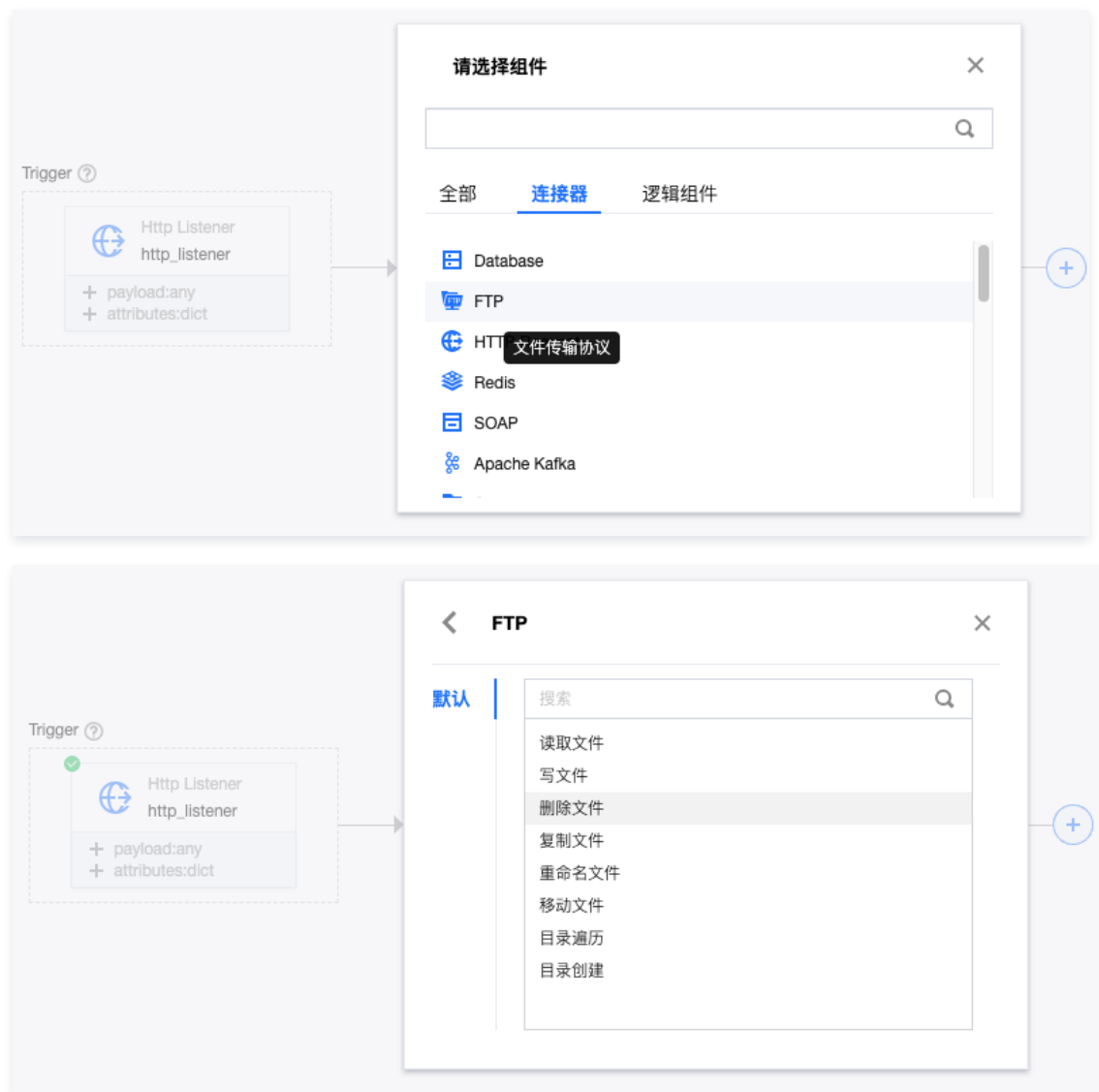
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/ftpctest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 删除文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的FTP服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方**测试连接**校验连接配置参数有效性。具体如下图：

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

通用

连接配置

域名

007.3vftp.com

端口号

int

21

用户名

victorwsw

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数，文件删除路径设为 `/data/ftpctest.txt`，参数输入见下图：

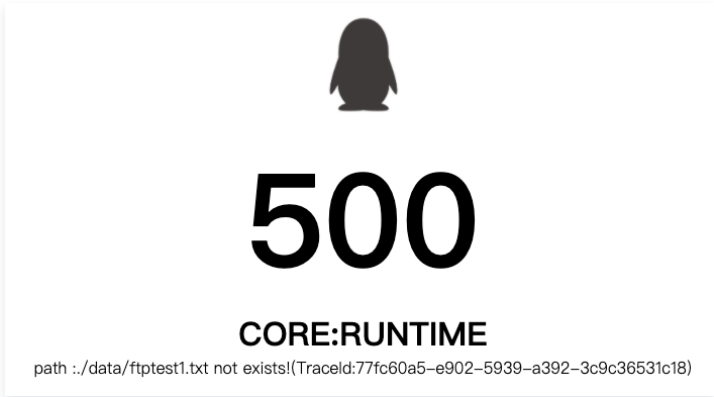
4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

○ FTP 删除文件操作，若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，FTP 输出消息 error 中会设置详细返错信息。



复制文件

输入参数

参数	数据类型	描述	是否必填	默认值
源文件路径	string	要复制文件的源路径支持复制单个文件和目录（包括目录下所有文件），支持表达式输入	是	-
目标文件路径	string	文件要复制到的目标路径支持表达式输入	是	-
创建父级目录	bool	是否创建不存在的父级目录	否	true
覆盖文件	bool	是否覆盖文件。当目标路径已存在同名源文件路径时，若 Overwrite 为 false，则抛错；若 Overwrite 为 true，则覆盖写	否	false
重命名	string	复制文件的新名称。如果未提供，则保留原始文件名；当目标路径已存在同名源文件路径时，可设置 RenameTo 参数命名新名称解决复制文件同名冲突支持表达式输入	否	-

FTP 复制文件

...

当前使用的是 ftptestpreCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

源文件路径

string

请输入字符串

f(x)

目标文件路径

string

请输入字符串

f(x)

创建父级目录

bool

true

覆盖文件

bool

false

重命名

string

请输入字符串

f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 复制文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

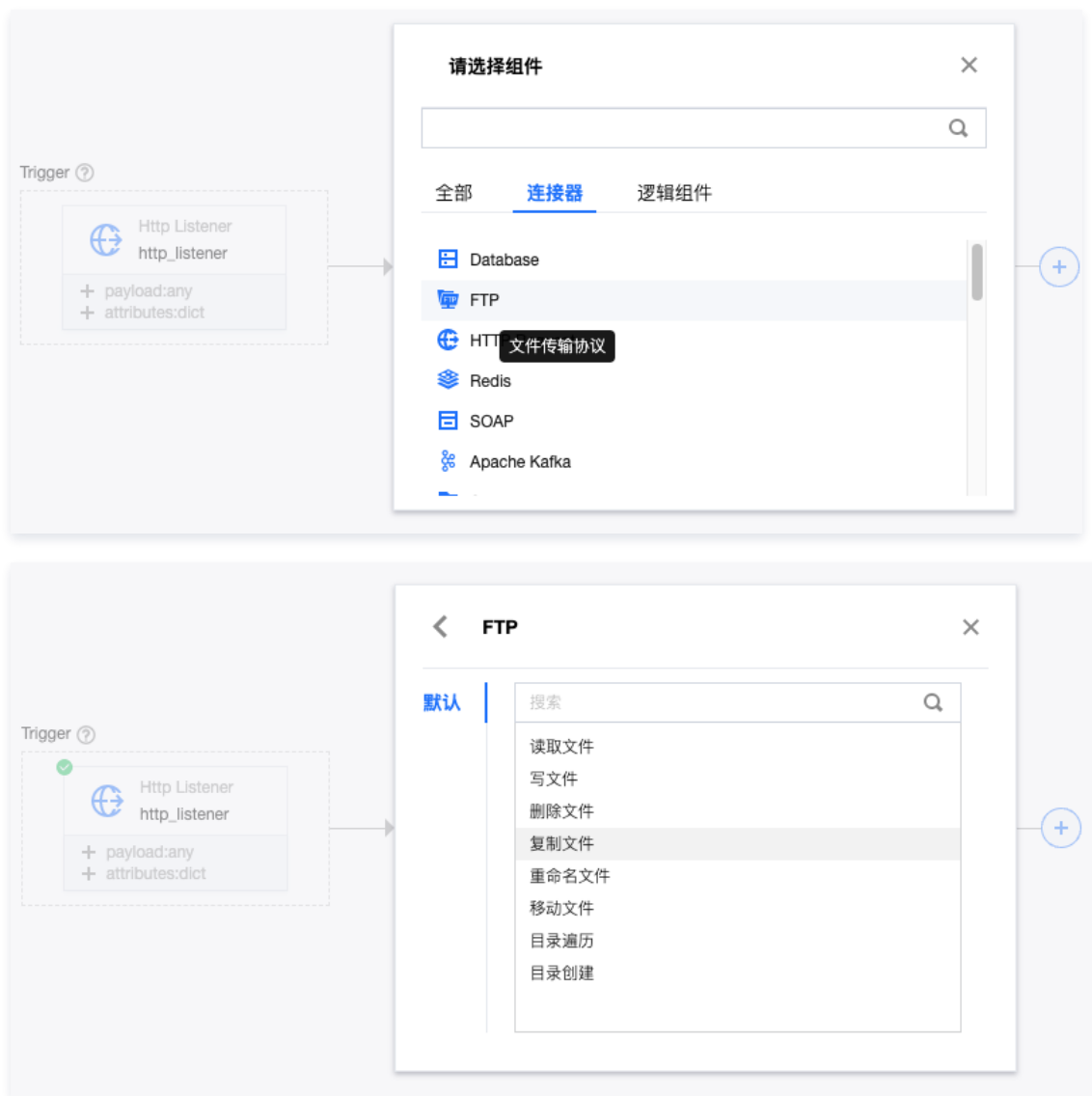
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/ftptest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 复制文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

2 配置连接器

通用

连接配置

域名

007.3vftp.com

端口号

int

21

用户名

victorwsw

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

- 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。源文件路径设为“/data/ftptest.txt”，目标文件路径设为“/data/copydir”，重命名设为“ftptestcopy.txt”，参数输入见下图：

FTP ? 复制文件

...

当前使用的是 ftptestpreCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

▲

源文件路径 *

string ▼

/data/ftptest.txt

f(x)

目标文件路径 *

string ▼

/data/copydir

f(x)

创建父级目录

bool ▼

true

▼

覆盖文件

bool ▼

false

▼

重命名

string ▼

ftptestcopy.txt

f(x)

高级配置

▲

输出消息绑定自定义数据类型

消息属性

payload ▼

类型选择

空 ▼

+

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存

调试

发布

删除

保存

请选择发布地域

☒ 天津

确定 取消

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

□

当前使用的是 ftp1 物理配置

查看

配置

备注

基本配置

▲

监听路径 *

/ftp1

指定监听的路径

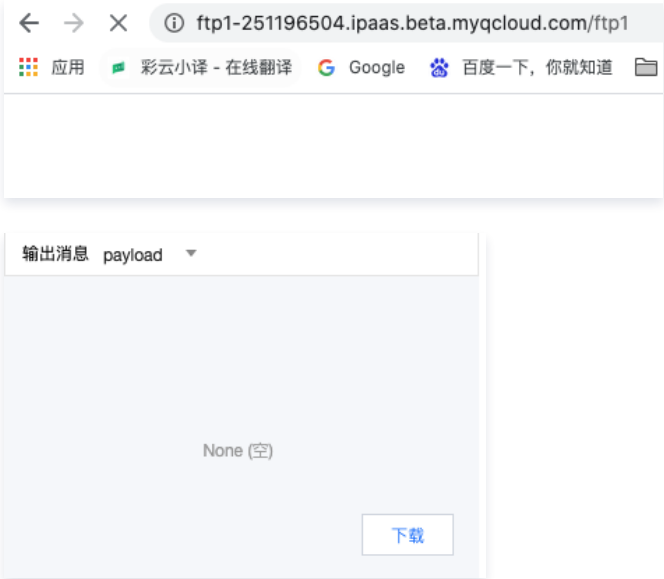
复制监听域名

监听方法 *

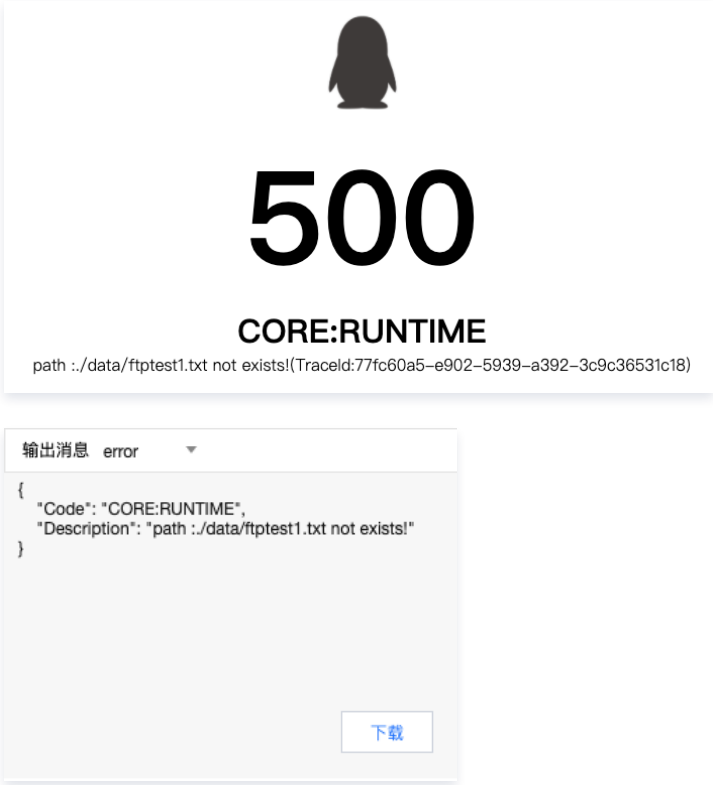
全部 ▼

指定监听HTTP请求的方法

- FTP 复制文件操作，若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，FTP 输出消息 error 中会设置详细返错信息。



重命名文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件路径	string	待重命名的文件路径： - 当路径或文件不存在时报错 - 同时支持目录和文件重命名，支持表达式输入	是	-
重命名名称	string	文件或路径的新名称支持表达式输入	是	-

覆盖文件	bool	是否覆盖文件： - 当 Overwrite 为 false 时，若重命名文件路径已存在则抛错 - 当 Overwrite 为 true 时，则直接覆盖写	否	false
------	------	--	---	-------

FTP 重命名文件

当前使用的是 ftptestpreCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

文件路径

string

请输入字符串

f(x)

重命名名称

string

请输入字符串

f(x)

覆盖文件

bool

false

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 重命名文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

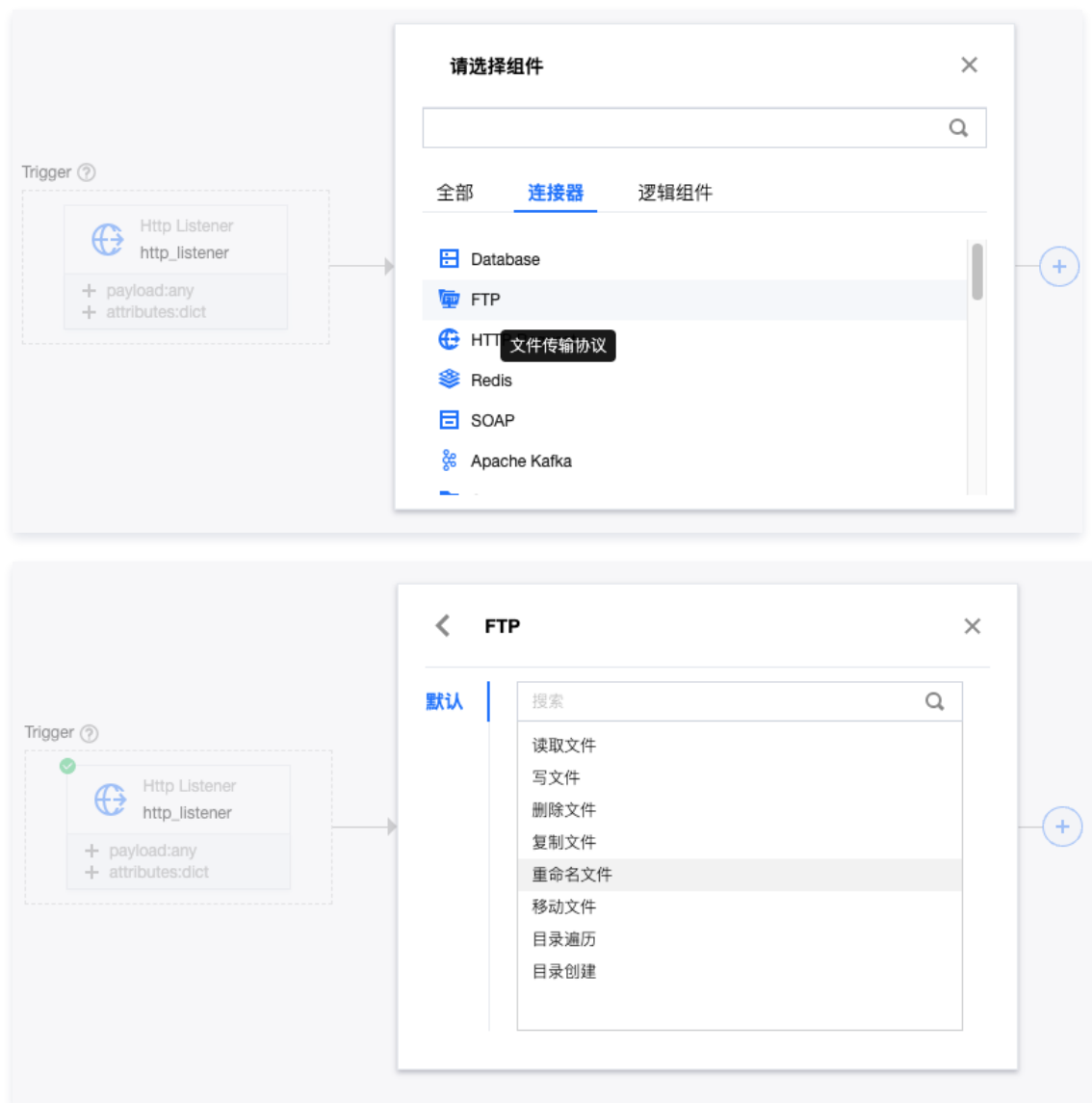
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/ftptest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 重命名文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

2 配置连接器

通用

连接配置

域名

007.3vftp.com

端口号

int

21

用户名

victorwsw

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

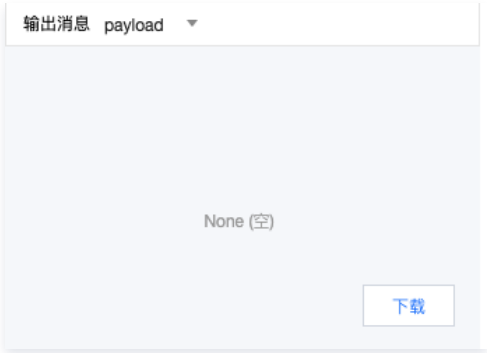
3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/ftptest.txt”，重命名名称设为“ftptestrename.txt”，参数输入见下图：

4. 操作配置参数设置完成后保存返回即可，然后单击右上角发布，选择发布地域后单击确定。

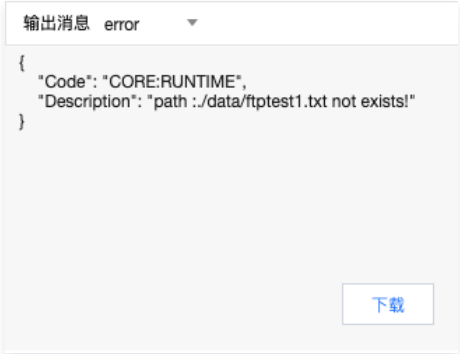
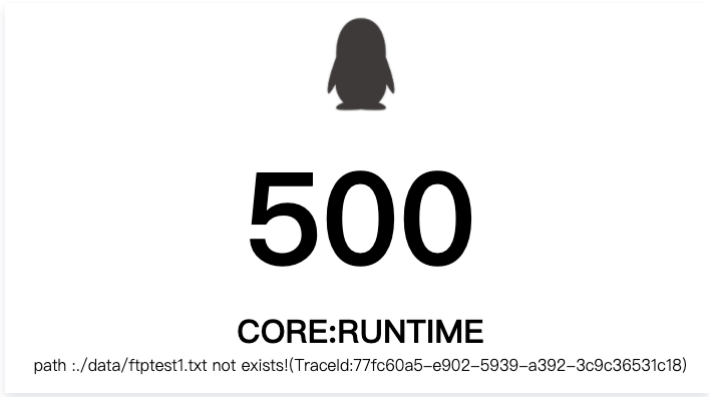
5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

○ FTP 重命名文件操作，若执行成功则返回 payload 响应结果为空。





- 当操作执行失败时，FTP 输出消息 error 中会设置详细返错信息。



移动文件

输入参数

参数	数据类型	描述	是否必填	默认值
源文件路径	string	要转移的文件或目录路径支持移动单个文件和目录（包括目录下所有文件），支持表达式输入	是	-
目标文件路径	string	文件要移动到的目标路径支持表达式输入	是	-
创建父级目录	bool	是否创建不存在的父级目录	否	true
覆盖文件	bool	是否覆盖文件。当目标路径已存在同名源文件路径时，若 Overwrite 为 false，则抛错；若 Overwrite 为 true，则覆盖写	否	false
重命名	string	移动文件的新名称。如果未提供，则保留原始文件名；当目标路径已存在同名源文件路径时，可设置 RenameTo 参数命名新名称解决移动文件同名冲突支持表达式输入	否	-

FTP 移动文件

当前使用的是 `ftptestpreCfg` 物理配置

编辑 | 切换

配置

调试

备注

通用

源文件路径

string

请输入字符串

f(x)

目标文件路径

string

请输入字符串

f(x)

创建父级目录

bool

true

覆盖文件

bool

false

重命名

string

请输入字符串

f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 移动文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

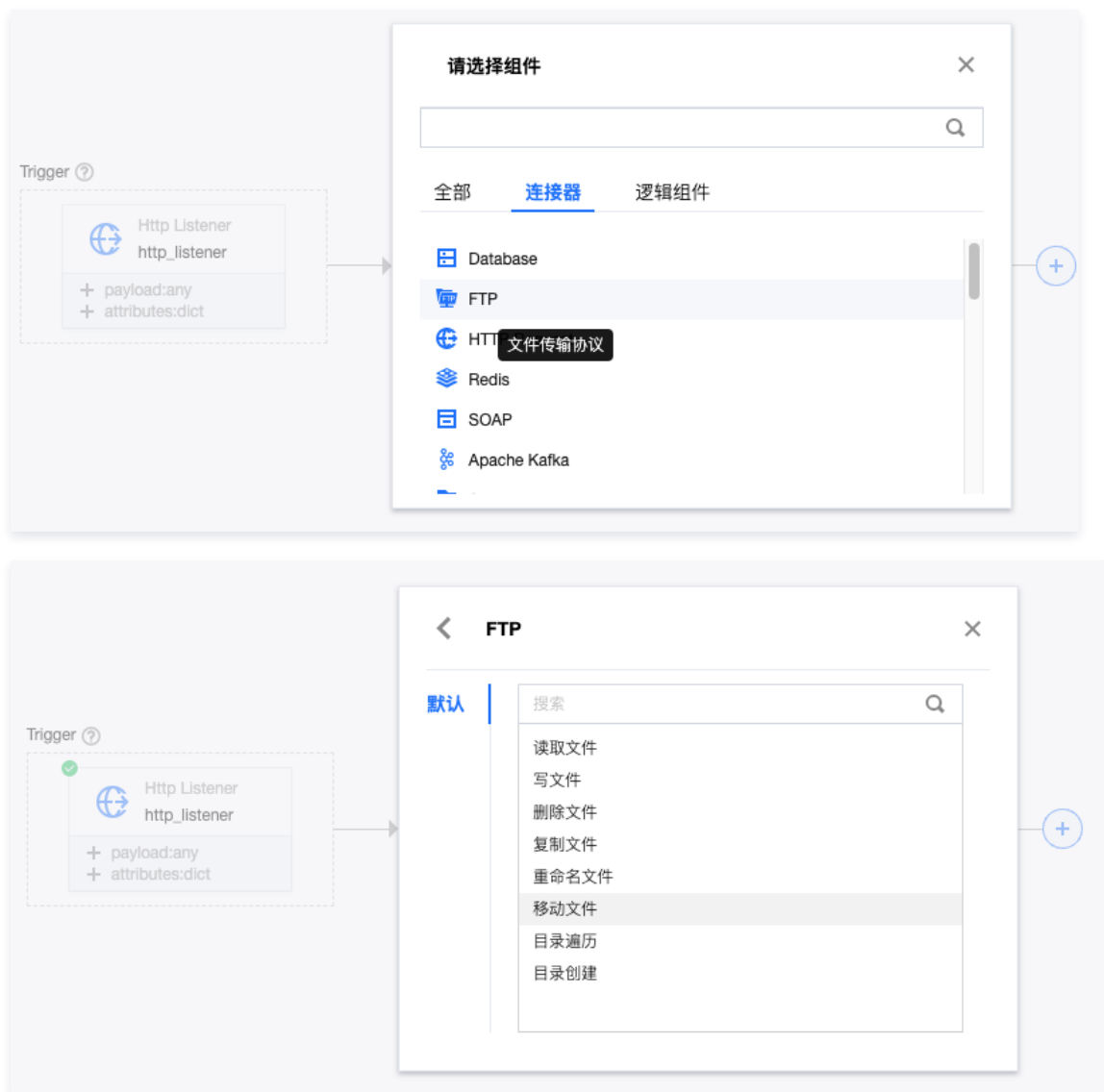
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/ftptest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 移动文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

2 配置连接器

通用

连接配置

域名

007.3vftp.com

端口号

int

21

用户名

victorwsw

密码

通用配置

工作路径

/

传输模式

BINARY

远程Host连接验证

bool

true

高级

重连接配置

重连接频率(秒)

int

10

通用配置

控制编码

ISO-8859-1

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

测试连接

上一步

保存

- 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。源文件路径设为“/data/ftptest.txt”，目标文件路径设为“/data/movedir”，重命名设为“ftptestmove.txt”，参数输入见下图：

FTP ? 移动文件

当前使用的是 `ftp1testpreCfg` 物理配置

编辑 | 切换

配置 调试 备注

通用

源文件路径

string /data/ftp1test.txt f(x)

目标文件路径

string /data/movedir f(x)

创建父级目录

bool true

覆盖文件

bool false

重命名

string ftp1testmove.txt f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空 +

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存 调试 发布 删除 保存

请选择发布地域

☒ 天津

确定 取消

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 `ftp1` 物理配置

查看

配置 备注

基本配置

监听路径

/ftp1

指定监听的路径

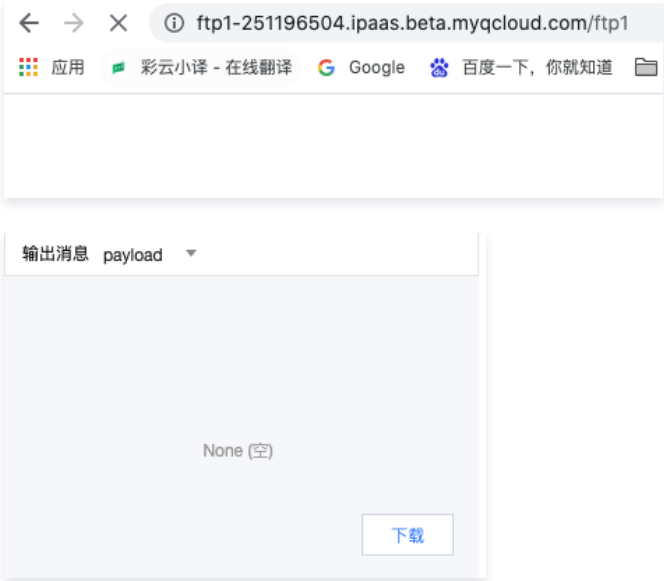
复制监听域名

监听方法

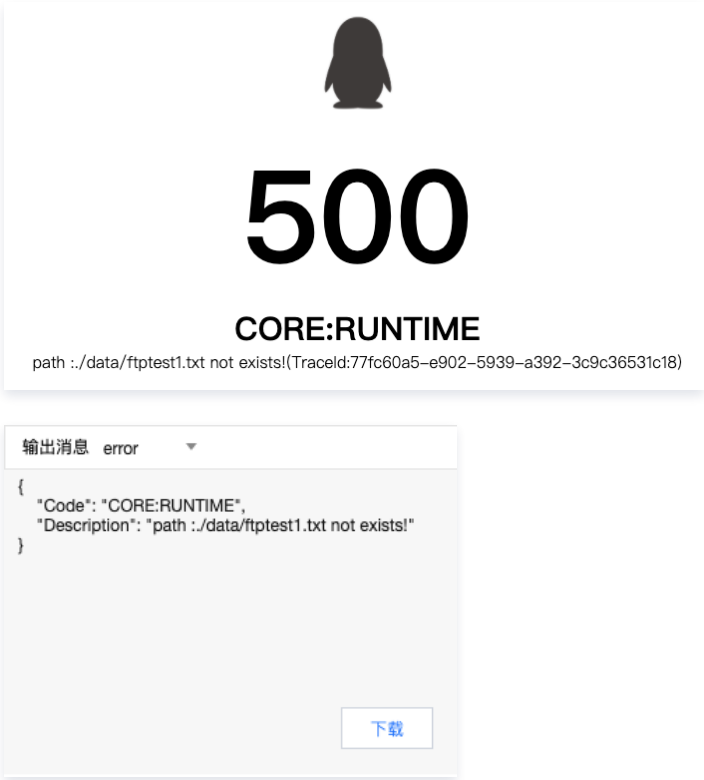
全部

指定监听HTTP请求的方法

- FTP 移动文件操作，若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，FTP 输出消息 error 中会设置详细报错信息。



目录遍历

输入参数

参数	数据类型	描述	是否必填	默认值
遍历目录路径	string	要列出的目录的路径支持表达式输入	是	
子目录递归轮询	bool	是否递归包含子目录的文件和目录	否	false

起始时间	string	在此日期之前创建的文件将被过滤，2020-11-14T08:30:00+00:00支持表达式输入	否	
截止时间	string	在此日期之后创建的文件将被过滤，2020-11-15T17:30:00+00:00支持表达式输入	否	
文件名正则匹配	String	使用正则表达式过滤文件或目录名称 [a-z]* 支持表达式输入	否	
文件路径匹配	String	目录路径过滤匹配模式，只保留指定文件或目录 /mydir/ftptest.txt 支持表达式输入	否	
目录	bool	是否过滤目录默认为 true，表示遍历结果中包含目录类型	否	true
文件	bool	是否过滤文件默认为 true，表示遍历结果中包含文件类型	否	true
符号链接	bool	是否过滤符号链接默认为 true，表示遍历结果中包含符号链接类型	否	true
最小比特值	int	文件 Size 过滤最小值	否	
最大比特值	int	文件 Size 过滤最大值	否	
文件校验时间间隔（秒）	int	文件大小校验等待时间（秒），范围：0~60，以判断文件是否可读。在等待时间前后执行两次 sizecheck 文件大小检测，如果两次检测都返回相同的值，即 sizecheck 期间文件没有其他修改操作则可以读取该文件。	否	0(s)
时间戳使能	bool	判断是否支持根据文件的修改或创建时间戳来判断触发流，缓存持久化记录上次轮询截止时间戳作为下次轮询起始判断时间。	否	false
时间戳状态标识	String	当时间戳使能为 true 时，需设置非空值作为当前流 FTP 组件时间戳状态缓存持久化的唯一识别标志。	否	

FTP 目录遍历

...

当前使用的是 ftpyouhuacfg 物理配置 [编辑](#) | [切换](#)

配置

备注

通用

遍历目录路径

string 请输入字符串 f(x)

子目录递归轮询

bool false

文件匹配配置

起始时间

string 2000-01-01 00:00:00 f(x)

截止时间

string 2000-01-01 00:00:00 f(x)

文件名正则匹配

string 请输入字符串 f(x)

文件路径匹配

string 请输入字符串 f(x)

目录

bool true

文件

bool true

符号链接

bool true

最小比特值

int 0

最大比特值

int 0

高级

文件校验时间间隔(秒)

int 0

时间戳使能

bool true

时间戳状态标识

string 请输入字符串 f(x)

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

- FTP 目录遍历操作会在消息 payload 中返回每个文件的属性信息：

参数	数据类型	描述
Name	string	文件名称
Type	string	文件类型: File Folder SymLink
Target	string	SymLink 对应路径
Size	int	文件大小
Time	string	文件最后修改时间
PathName	string	文件绝对路径

输出消息 payload ▾

```

"copydir": {
  "PathName": "/data/copydir",
  "Size": 0,
  "Target": null,
  "Time": "2021-03-22 15:47:00",
  "Type": "Folder"
},
"ftptest.txt": {
  "PathName": "/data/ftptest.txt",
  "Size": 0,
  "Target": null,
  "Time": "2021-03-22 15:57:00",
  "Type": "File"
},
"movedir": {
  "PathName": "/data/movedir"
}
    
```

下载

- FTP 目录遍历操作，若执行失败时输出消息 error 中会设置详细报错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

输出消息 error ▾

```

{
  "Code": "CORE:RUNTIME",
  "Description": "FTP组件配置的directoryPath属性格式错误. directoryPath: @@/data, error: filePathPattern is not matched!"
}
    
```

下载

案例

- 在连接器列表中选择 FTP 连接器，然后选择 FTP 目录遍历操作。

Trigger ?

- Http Listener
- http_listener
- + payload:any
- + attributes:dict

请选择组件

全部 连接器 逻辑组件

- Database
- FTP
- HTTP 文件传输协议
- Redis
- SOAP
- Apache Kafka



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置



- ✓

选择连接器
- >
- 2

配置连接器

通用

连接配置

域名 *

端口号

int ▼

用户名

密码

通用配置

工作路径

传输模式

BINARY ▼

远程Host连接验证

bool ▼

true ▼

高级

重连接配置

重连接频率(秒)

int ▼

通用配置

控制编码

ISO-8859-1 ▼

超时配置

连接超时时间(秒)

int ▼

响应超时时间(秒)

int ▼

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。遍历目录路径设为“/data”，其他参数输入见下图：

FTP 目录遍历

当前使用的是 ftptestpreCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

遍历目录路径

string

/data

f(x)

子目录递归轮询

bool

false

文件匹配配置

起始时间

string

2000-01-01 00:00:00

f(x)

截止时间

string

2000-01-01 00:00:00

f(x)

文件名正则匹配

string

请输入字符串

f(x)

文件路径匹配

string

请输入字符串

f(x)

目录

bool

true

文件

bool

true

符号链接

bool

true

最小比特值

int

0

最大比特值

int

0

高级

文件校验时间间隔(秒)

int

0

时间戳使能

bool

false

时间戳状态标识

string

请输入字符串

f(x)

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存

调试

发布

删除

保存

请选择发布地域

天津

确定

取消

5. 待集成流发布成功后复制HTTP Listener监听路径后访问该域名即可触发流。

Http Listener

当前使用的是 ftp1 物理配置 [查看](#)

配置

备注

基本配置

监听路径

/ftp1

指定监听的路径

复制监听域名

监听方法

全部

指定监听HTTP请求的方法

○ FTP 返回 payload 响应结果如下，即为“/data”目录下遍历文件的属性列表。

```
{".":{"PathName":"/data/.","Size":0,"Target":"","Time":"2021-03-22 15:57:00","Type":"Folder"},".":{"PathName":"/data/.","Size":0,"Target":"","Time":"2021-03-22 15:57:00","Type":"Folder"},"copydir":{"PathName":"/data/copydir","Size":0,"Target":"","Time":"2021-03-22 15:47:00","Type":"Folder"},"ftptest.txt":{"PathName":"/data/ftptest.txt","Size":0,"Target":"","Time":"2021-03-22 15:57:00","Type":"File"},"movedir":{"PathName":"/data/movedir","Size":0,"Target":"","Time":"2021-03-22 15:56:00","Type":"Folder"}}
```

输出消息 payload

```
"copydir": {
  "PathName": "/data/copydir",
  "Size": 0,
  "Target": null,
  "Time": "2021-03-22 15:47:00",
  "Type": "Folder"
},
"ftptest.txt": {
  "PathName": "/data/ftptest.txt",
  "Size": 0,
  "Target": null,
  "Time": "2021-03-22 15:57:00",
  "Type": "File"
},
"movedir": {
  "PathName": "/data/movedir"
```

下载

○ 当操作执行失败时，FTP 输出消息 error 中会设置详细报错信息。

500

CORE:RUNTIME

FTP组件配置的directoryPath属性格式错误。 directoryPath: @/@/data, error: filePathPattern is not matched!(TraceId:4ba6d22d-ae02-3907-0dd2-15b811850858)

输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "FTP组件配置的directoryPath属性格式错误。 directoryPath: @/@/data, error: filePathPattern is not matched!"
}
```

下载

目录创建

输入参数

参数	数据类型	描述	是否必填
目录创建路径	string	新创建目录的名称默认支持创建多级目录，当路径已存在时返错支持表达式输入	是

FTP 目录创建

当前使用的是 ftptestpreCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

目录创建路径

string

请输入字符串

f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

FTP 目录创建操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细返错信息。错误输出为dict类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

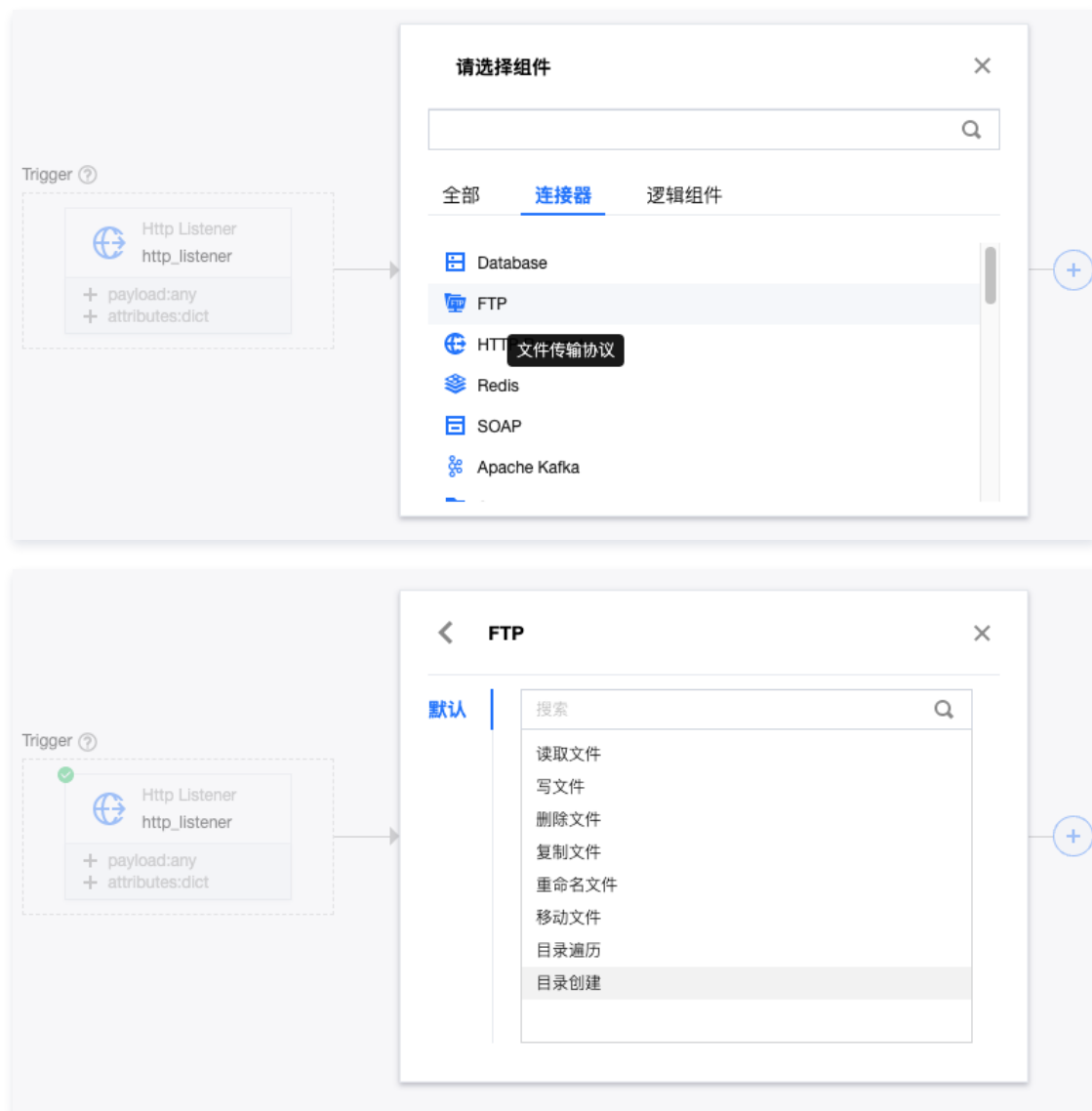
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/ftptest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 FTP 连接器，然后选择 FTP 目录创建操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 FTP 服务器对应参数，工作路径设置为 FTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方**测试连接**校验连接配置参数有效性。具体如下图：

编辑连接器配置



- ✓

选择连接器
- >
- 2

配置连接器

通用



连接配置

域名 *

端口号

int ▼

用户名

密码

通用配置

工作路径

传输模式

BINARY ▼

远程Host连接验证

bool ▼

true ▼

高级



重连接配置

重连接频率(秒)

int ▼

通用配置

控制编码

ISO-8859-1 ▼

超时配置

连接超时时间(秒)

int ▼

响应超时时间(秒)

int ▼

测试连接

上一步

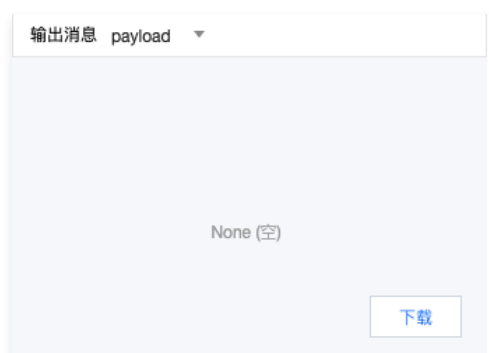
保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数，文件路径设为“/data/a/b/c”，参数输入见下图：

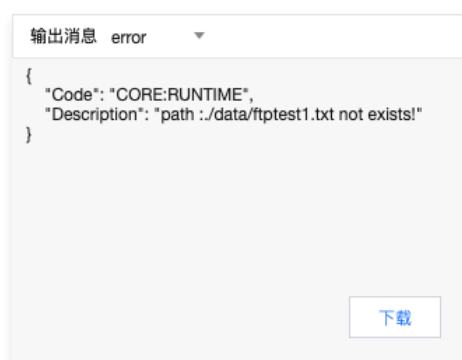
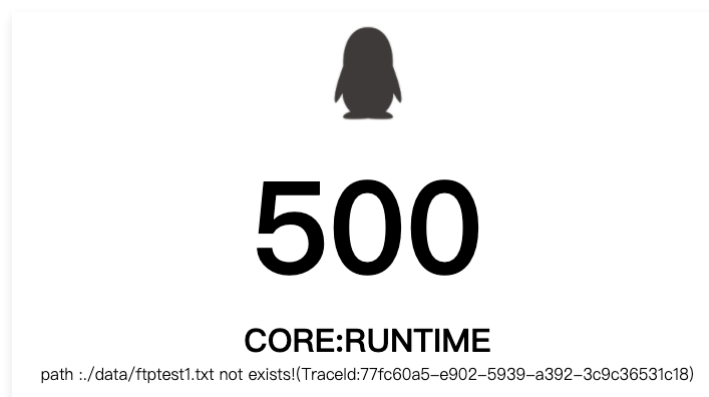
4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

○ FTP 目录创建操作，若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，FTP 输出消息 error 中会设置详细报错信息。



SFTP 连接器

最近更新时间：2024-10-18 16:14:52

简介

- SSH 文件传输协议（SSH File Transfer Protocol，也称 Secret File Transfer Protocol 即安全文件传送协议）是一 [数据流](#) 连接，提供文件访问、传输和管理功能的 [网络传输协议](#)。
- SSH（Secure Shell）：安全外壳协议，由 IETF 的网络小组（Network Working Group）所制定；SSH 为建立在应用层基础上的安全协议。SSH 是目前较可靠，专为远程登录会话和其他网络服务提供安全性的协议。利用 SSH 协议可以有效防止远程管理过程中的信息泄露问题。
- SFTP 与 FTP 有着几乎一样的语法和功能。SFTP 为 SSH 的其中一部分，是一种传输档案至 Blogger 服务器的安全方式。其实在 SSH 软件包中，已经包含一个叫作 SFTP 的安全文件信息传输子系统，SFTP 本身没有单独的守护进程，它必须使用 sshd 守护进程（端口号默认是22）来完成相应的连接和答复操作，所以从某种意义上来说，SFTP 并不像一个服务器程序，而更像是一个客户端程序。
- SFTP 与 FTP 的主要区别：
 - 连接方式：FTP 使用 TCP 端口21上的控制连接建立连接，而 SFTP 是在客户端和服务器之间通过 SSH 协议（TCP 端口22）建立的安全连接来传输文件。
 - 安全性：SFTP 使用加密传输认证信息和传输的数据，所以使用 SFTP 相对于 FTP 是非常安全。
 - 效率：SFTP 传输方式使用加密解密技术，所以传输效率比普通的 FTP 要低得多。
- SFTP（SFTP Connector）提供对 SFTP 服务器上的文件和文件夹的访问。SFTP 连接器操作通过 SFTP（安全文件传输协议）协议管理文件传输。连接器的主要特点包括：
 - 根据需要读取文件或列出目录内容。
 - 支持常见的 SFTP 操作，如创建目录、复制、移动、重命名、删除文件。
 - 支持锁定文件。
 - 文件匹配功能。

连接器配置

通用配置

参数	数据类型	描述	是否必填	默认值
域名	string	SFTP 服务器域名	是	-
端口号	int	SFTP 服务器端口号	否	22
用户名	string	SFTP 服务器的用户名。如果服务器需要身份验证，则为必需参数	否	-
密码	string	SFTP 服务器的密码。如果服务器需要身份验证，则为必需参数	否	-
工作路径	string	指定为此连接器所使用的每个相对路径的根目录。如果未提供，则默认为远程服务器默认值	否	/
SSH 公钥认证	bool	是否开启 SSH 公钥认证	否	false
私钥认证文件	string	私钥认证文件	否	-
认证文件密码	string	私钥认证文件密码。如果没有提供标识文件 Identity File，参数将被忽略	否	-
KNOWN_HOSTS 密钥验证	bool	是否开启 knownHosts 密钥验证	否	false
KNOWN_HOSTS 密钥验证文件	string	如果已提供，客户机将根据引用文件中的 knownHosts 密钥验证服务器的密钥。如果服务器密钥与文件中的密钥不匹配，连接将被中止	否	-

高级配置

参数	数据类型	描述	是否必填	默认值
代理域名	string	代理服务器域名	否	-
代理端口号	int	代理服务器端口号	否	-
代理用户名	string	代理服务器用户名	否	-

代理密码	string	代理服务器密码	否	-
代理协议	enum	代理服务器协议：HTTP、SOCKS4、SOCKS5	否	-
重连接频率（秒）	int	重连接频率（单位s）（范围：0~300）	否	10（s）
连接超时时间（秒）	int	连接超时等待时间（单位s）（范围：0~300）	否	60（s）
响应超时时间（秒）	int	请求响应超时等待时间（单位s）（范围：0~300）	否	60（s）

连接器配置界面如下：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

域名

请输入字符串

端口号

int

22

用户名

请输入字符串

密码

工作路径

/

SSH配置

SSH公钥认证

bool

false

KNOWN_HOST
S密钥验证

bool

false

高级

代理配置

代理域名

请输入字符串

代理端口号

int

请输入整数

代理用户名

请输入字符串

代理密码

请输入字符串

代理协议

请选择

重连接配置

重连接频率(秒)

int

10

超时配置

连接超时时间
(秒)

int

60

响应超时时间
(秒)

int

60

安全网关

安全网关

无

测试连接

上一步

保存

操作配置

读取文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件路径	string	读文件路径	是	
文件锁	bool	是否开启文件锁	否	false
文件锁等待时间（秒）	int	文件锁等待时间（秒）（范围：0~60）	否	30（s）
文件校验时间间隔（秒）	int	文件大小校验等待时间（秒），以判断文件是否可读。在等待时间前后执行两次 sizecheck 文件大小检测。如果两次检测都返回相同的值，即 sizecheck 期间文件没有其他修改操作则可以读取该文件（范围：0~60）	否	0（s）

SFTP 读取文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

文件路径

string 请输入字符串 f(x)

高级

文件锁

bool false

文件锁等待时间 (秒)

int 30

文件校验时间间隔(秒)

int 0

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

 +

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 Entity 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

- SFTP 读取文件操作会返回文件内容，根据文件名后缀设置消息的 mimeType 和 encoding 并放到消息 payload 中，后续可通过 dataway 表达式直接访问。

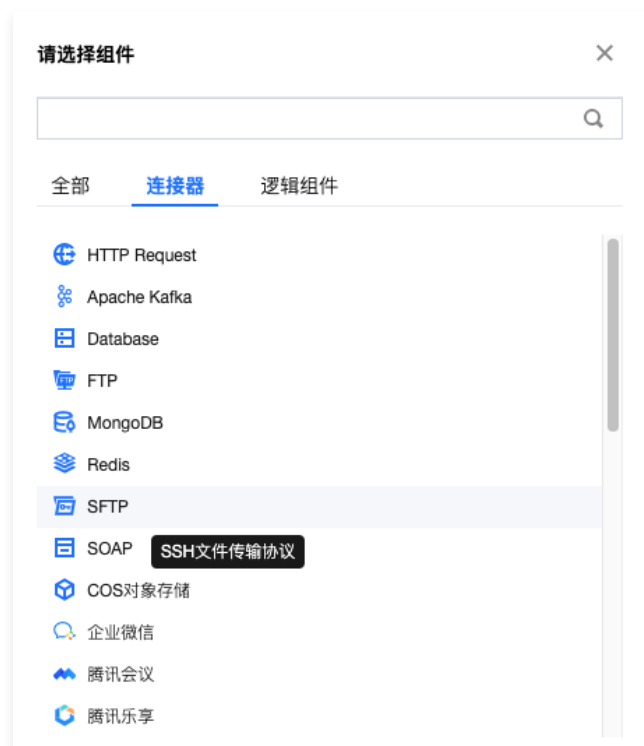


- SFTP 读取文件操作若执行失败时输出消息 error 中会设置详细报错信息。错误输出为 dict 类型，包含 “Code” 和 “Description” 元素，“Code” 表示错误类型，“Description” 表示错误具体信息，例如：



案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 读取文件操作。





2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/” ，其他参数保持默认值即可。参数填写完后可单击下方**测试连接**校验连接配置参数有效性。具体如下图：

编辑连接器配置



1 选择连接器 > 2 配置连接器

通用

连接配置

域名 *

端口号

用户名

密码

工作路径

SSH配置

SSH公钥认证

KNOWN_HOST S密钥验证

高级

代理配置

代理域名

代理端口号

代理用户名

代理密码

代理协议

重连接配置

重连接频率(秒)

超时配置

连接超时时间(秒)

响应超时时间(秒)

安全网关

安全网关

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/sftpptest.txt”，参数输入见下图：

SFTP 读取文件

当前使用的是 SFTPProdCfg 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

通用

文件路径 /data/sftpptest.txt

高级

文件锁 false

文件锁等待时间 (秒) 30

文件校验时间间隔 (秒) 0

高级配置

输出消息绑定自定义数据类型

消息属性

类型选择

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存 [调试](#) [发布](#) [删除](#) [保存](#)

请选择发布地域

☒ 天津

[确定](#) [取消](#)

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener

当前使用的是 SFTP1 物理配置 [查看](#)

配置 备注

基本配置

监听路径

指定监听的路径

监听方法

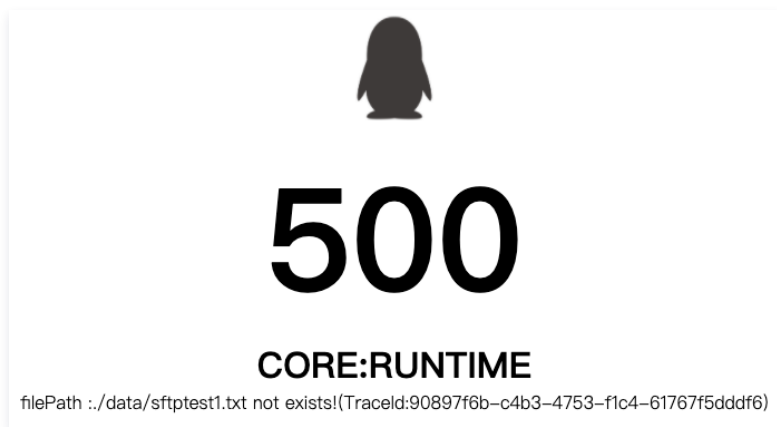
指定监听HTTP请求的方法

复制监听域名

- SFTP 返回 payload 响应结果如下，即为 “/data/sftpctest.txt” 文件内容。



- 当操作执行失败时，SFTP 输出消息 error 中会设置详细报错信息。



写文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件路径	string	待写入文件路径（当检测为目录时抛错）支持表达式输入	是	-
文件内容	string	待写入文件的内容。默认为当前消息 payload 支持表达式输入	否	#[payload]
创建父级目录	bool	是否尝试创建任何不存在的父目录	否	true

文件锁	bool	是否锁定文件	否	false
文件锁等待时间（秒）	int	文件锁等待时间（秒）（范围：0~60）	否	30（s）
文件写模式	enum: OVERWRITE、 APPEND、 CREATE_NEW	文件写模式： - OVERWRITE：覆盖写 - APPEND：追加写 - CREATE_NEW：当文件不存在时新建，否则抛错	否	OVERWRITE

SFTP 写文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

调试

备注

通用

文件路径

string

请输入字符串

f(x)

文件内容

string

请输入字符串

f(x)

创建父级目录

bool

true

文件锁

bool

false

文件锁等待时间(秒)

int

30

文件写模式

OVERWRITE

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

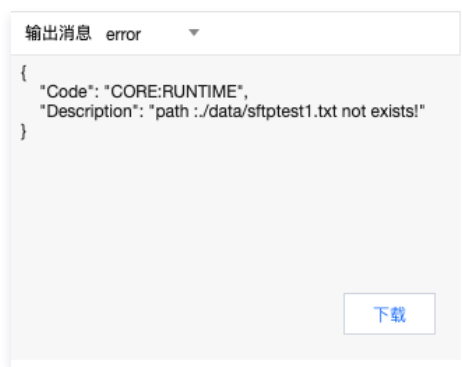
+

输出

组件输出的 message 信息如下：

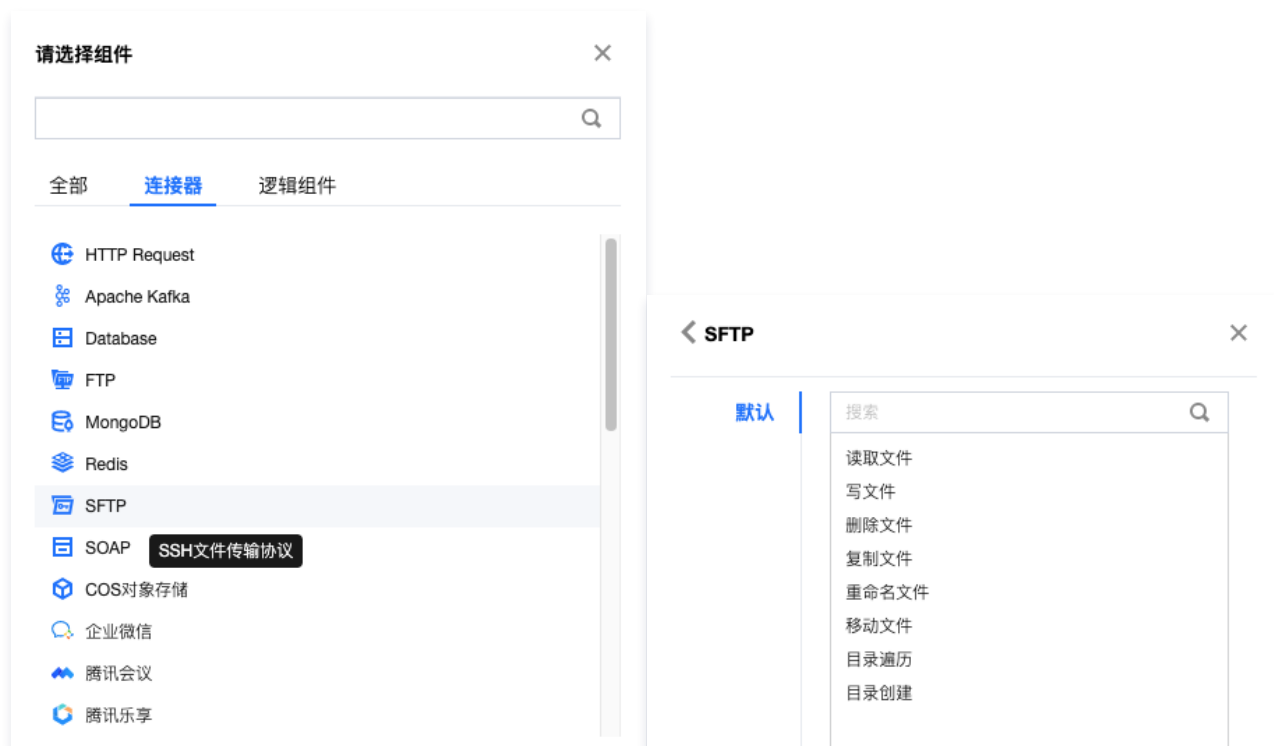
message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 写文件操作，若执行成功则 payload 为空；若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：



案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 写文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置



1 选择连接器 > 2 配置连接器

通用

连接配置

域名 *

端口号

用户名

密码

工作路径

SSH配置

SSH公钥认证

KNOWN_HOST S密钥验证

高级

代理配置

代理域名

代理端口号

代理用户名

代理密码

代理协议

重连接配置

重连接频率(秒)

超时配置

连接超时时间 (秒)

响应超时时间 (秒)

安全网关

安全网关

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/sftpptest.txt”，参数输入见下图：

SFTP 写文件

当前使用的是 SFTPProdCfg 物理配置 [编辑](#) | [切换](#)

配置 调试 备注

通用

文件路径 /data/sftpptest.txt

文件内容 hello world

创建父级目录 true

文件锁 false

文件锁等待时间 (秒) 30

文件写模式

高级配置

输出消息绑定自定义数据类型

消息属性

类型选择

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存 [调试](#) [发布](#) [删除](#) [保存](#)

请选择发布地域

☒ 天津

[确定](#) [取消](#)

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener

当前使用的是 SFTP1 物理配置 [查看](#)

配置 备注

基本配置

监听路径

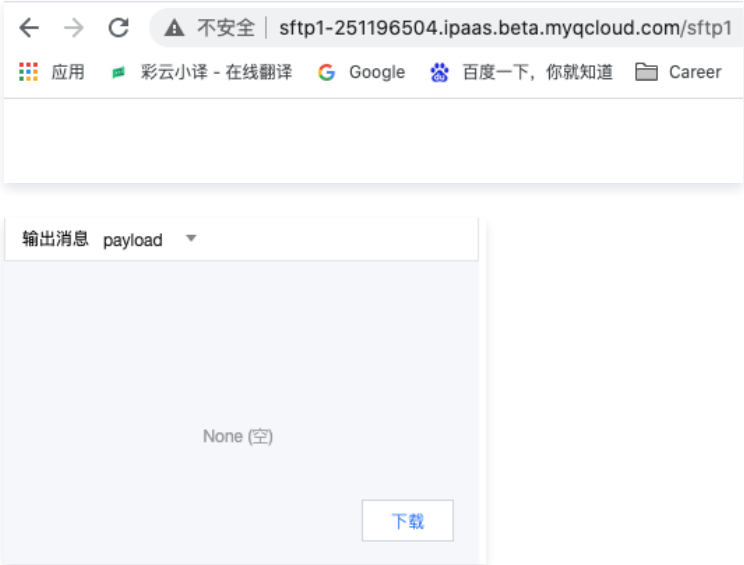
指定监听的路径

监听方法

指定监听HTTP请求的方法

复制监听域名

- SFTP 写文件操作若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，SFTP 输出消息 error 中会设置详细返错信息。



删除文件

输入参数

参数	数据类型	描述	是否必填
文件删除路径	string	待删除的文件或路径： - 当文件或路径不存在时抛错 - 支持删除单个文件以及目录（包括目录下所有文件） - 如果文件没有被锁定，则删除路径指向的文件，支持表达式输入	是

SFTP 删除文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

文件删除路径

string 请输入字符串 f(x)

高级配置

输出消息绑定自定义数据类型

消息属性 payload

类型选择 空 +

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 删除文件操作若执行成功 payload 为空，若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

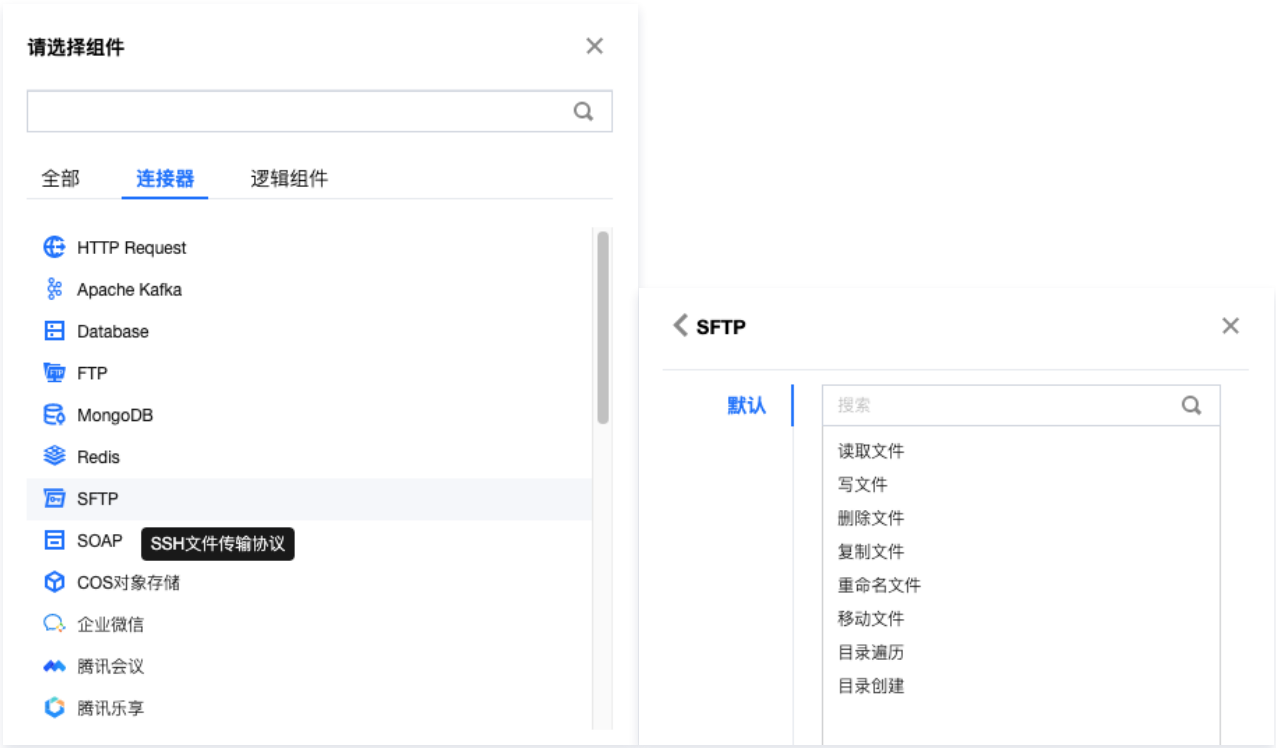
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/sftpctest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 删除文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

域名

159.75.18.194

端口号

int

2255

用户名

victorwsw

密码

工作路径

/

SSH配置

SSH公钥认证

bool

false

KNOWN_HOST
S密钥验证

bool

false

高级

代理配置

代理域名

请输入字符串

代理端口号

int

请输入整数

代理用户名

请输入字符串

代理密码

请输入字符串

代理协议

请选择

重连接配置

重连接频率(秒)

int

10

超时配置

连接超时时间
(秒)

int

60

响应超时时间
(秒)

int

60

安全网关

安全网关

无

测试连接

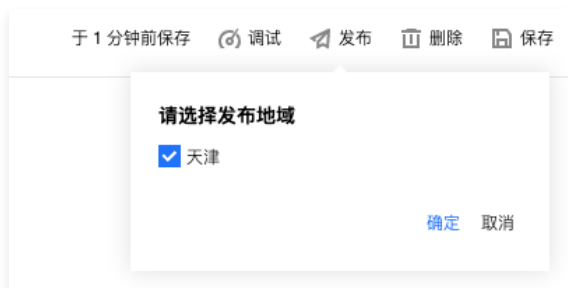
上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件删除路径设为“/data/sftpptest.txt”，参数输入见下图：



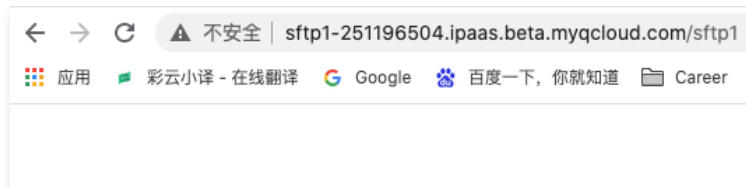
4. 操作配置参数设置完成后保存返回即可，单击右上角**发布**，选择发布地域后单击**确定**。



5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

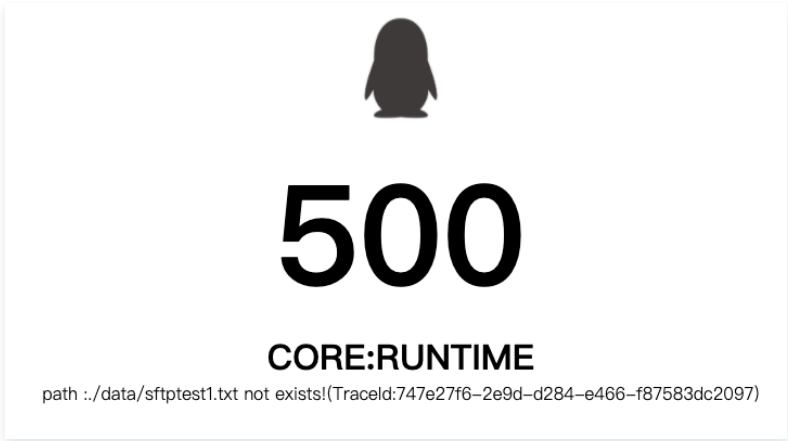


○ SFTP 删除文件操作若执行成功则返回 payload 响应结果为空。





- 当操作执行失败时，SFTP 输出消息 error 中会设置详细报错信息。



复制文件

输入参数

参数	数据类型	描述	是否必填	默认值
源文件路径	string	要复制文件的源路径支持复制单个文件和目录（包括目录下所有文件）支持表达式输入	是	-
目标文件路径	string	文件要复制到的目标路径支持表达式输入	是	-
创建父级目录	bool	是否创建不存在的父级目录	否	true
覆盖文件	bool	是否覆盖文件，当目标路径已存在同名源文件路径时，若 Overwrite 为 false 则抛错，若 Overwrite 为 true 则覆盖写	否	false
重命名	string	复制文件的新名称，如果未提供，则保留原始文件名。当目标路径已存在同名源文件路径时，可设置 RenameTo 参数命名新名称解决复制文件同名冲突支持表达式输入	否	-

SFTP 复制文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

源文件路径

string

请输入字符串

f(x)

目标文件路径

string

请输入字符串

f(x)

创建父级目录

bool

true

覆盖文件

bool

false

重命名

string

请输入字符串

f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 复制文件操作若执行成功 payload 为空，若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

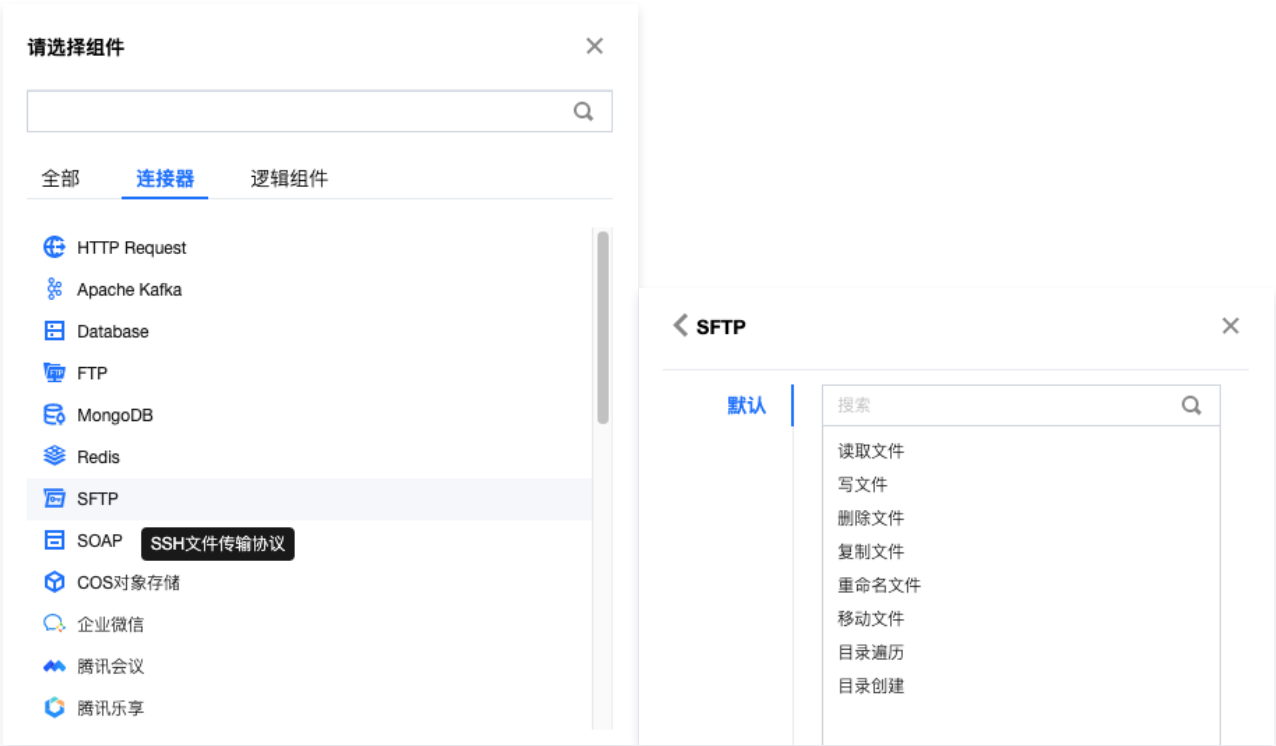
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/sftpctest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 复制文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

域名

159.75.18.194

端口号

int

2255

用户名

victorwsw

密码

工作路径

/

SSH配置

SSH公钥认证

bool

false

KNOWN_HOST

bool

false

S密钥验证

高级

代理配置

代理域名

请输入字符串

代理端口号

int

请输入整数

代理用户名

请输入字符串

代理密码

请输入字符串

代理协议

请选择

重连接配置

重连接频率(秒)

int

10

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

安全网关

安全网关

无

测试连接

上一步

保存

- 确认连接器配置参数填写无误后单击保存，然后设置操作配置参数。源文件路径设为“/data/sftpctest.txt”，目标文件路径设为“/data/copydir”，重命名设为“sftpctestcopy.txt”，参数输入见下图：

版权所有：腾讯云计算（北京）有限责任公司

第529 共732页

SFTP ? 复制文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置 备注

通用

源文件路径 *

string /data/sftpctest.txt f(x)

目标文件路径 *

string /data/copydir f(x)

创建父级目录

bool true

覆盖文件

bool false

重命名

string sftpctestcopy.txt f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空 +

4. 操作配置参数设置完成后保存返回即可，单击右上角发布，选择发布地域后单击确定。

于 1 分钟前保存 调试 发布 删除 保存

请选择发布地域

☒ 天津

确定 取消

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 SFTP1 物理配置

查看

配置 备注

基本配置

监听路径 *

/sftp1

指定监听的路径

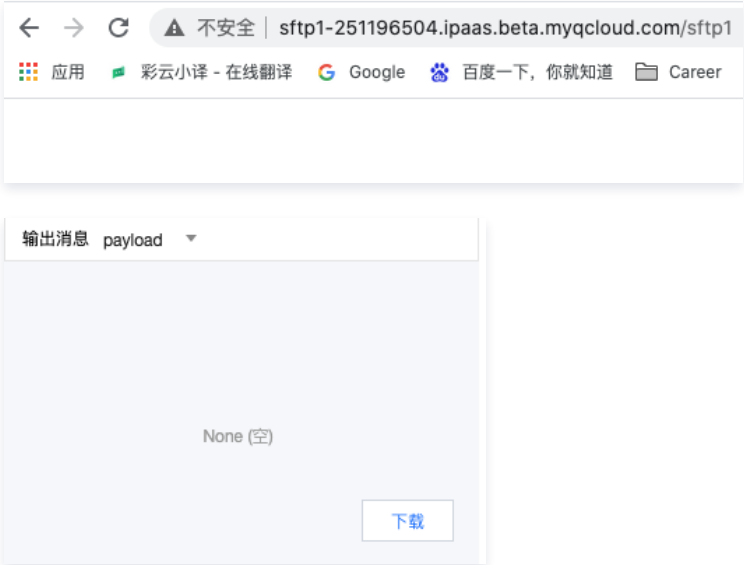
复制监听域名

监听方法 *

全部

指定监听HTTP请求的方法

- SFTP 复制文件操作若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，SFTP 输出消息 error 中会设置详细报错信息。



重命名文件

输入参数

参数	数据类型	描述	是否必填	默认值
文件路径	string	待重命名的文件路径： - 当路径或文件不存在时报错 - 同时支持目录和文件重命名,支持表达式输入	是	-

重命名名称	string	文件或路径的新名称,支持表达式输入	是	-
覆盖文件	bool	是否覆盖文件： - 当 Overwrite 为 false 时，若重命名文件路径已存在则抛错 - 当 Overwrite 为 true 时则直接覆盖写	否	false

SFTP 重命名文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

文件路径

string

请输入字符串

f(x)

重命名名称

string

请输入字符串

f(x)

覆盖文件

bool

false

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 重命名文件操作若执行成功则 payload 为空，若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

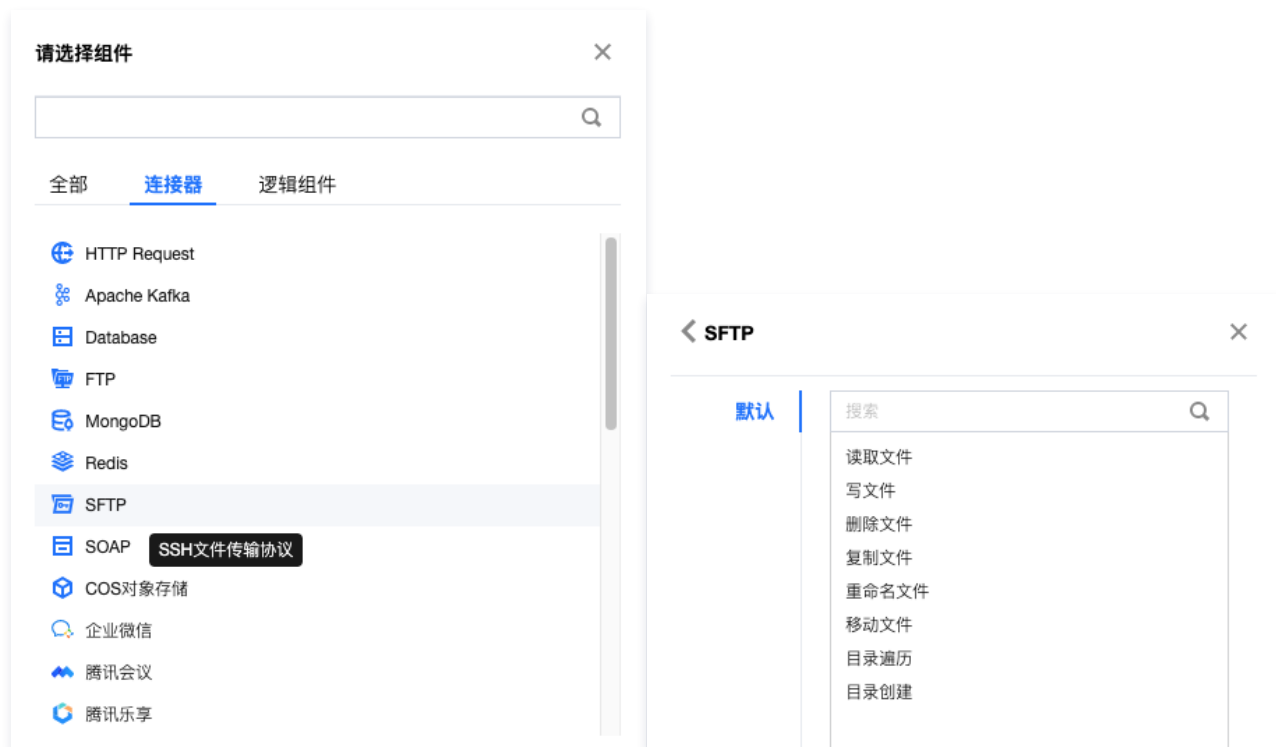
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path :./data/sftpctest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 重命名文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径“/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

域名

159.75.18.194

端口号

int

2255

用户名

victorwsw

密码

工作路径

/

SSH配置

SSH公钥认证

bool

false

KNOWN_HOST

bool

false

S密钥验证

高级

代理配置

代理域名

请输入字符串

代理端口号

int

请输入整数

代理用户名

请输入字符串

代理密码

请输入字符串

代理协议

请选择

重连接配置

重连接频率(秒)

int

10

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

安全网关

安全网关

无

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/sftpctest.txt”，重命名名称设为“sftpctestrename.txt”，参数输入见下图：

SFTP ? 重命名文件

当前使用的是 sftpbetatestCfg 物理配置
[编辑](#) | [切换](#)

配置备注

通用

文件路径 *

string /data/sftpctest.txt f(x)

重命名名称 *

string sftpctestrename.txt f(x)

覆盖文件

bool false

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空 +

4. 操作配置参数设置完成后保存返回即可，单击右上角发布，选择发布地域后单击确定。

于 1 分钟前保存
调试
发布
删除
保存

请选择发布地域

☒ 天津

[确定](#)
[取消](#)

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 SFTP1 物理配置
[查看](#)

配置备注

基本配置

监听路径 *

/sftp1

指定监听的路径

复制监听域名

监听方法 *

全部

指定监听HTTP请求的方法

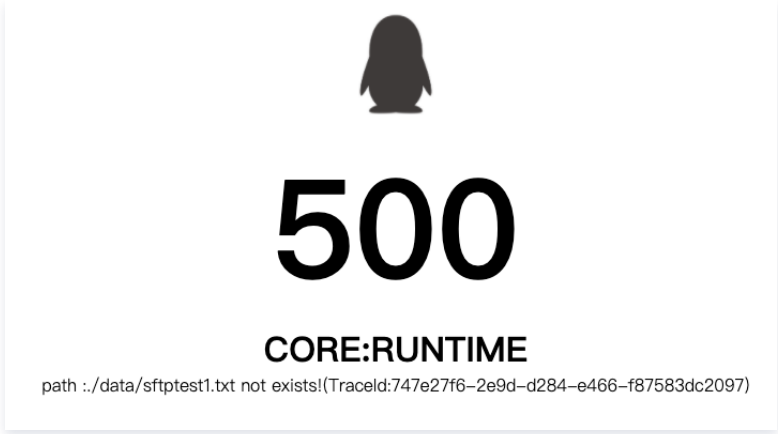
○ SFTP 重命名文件操作若执行成功则返回 payload 响应结果为空。

← → ↻ ⚠ 不安全 | sftp1-251196504.ipaas.beta.myqcloud.com/sftp1

应用 彩云小译 - 在线翻译 Google 百度一下，你就知道 Career



- 当操作执行失败时，SFTP 输出消息 error 中会设置详细返错信息。



移动文件

输入参数

参数	数据类型	描述	是否必填	默认值
源文件路径	string	要转移的文件或目录路径支持移动单个文件和目录（包括目录下所有文件）支持表达式输入	是	-
目标文件路径	string	文件要移动到的目标路径支持表达式输入	是	-
创建父级目录	bool	是否创建不存在的父级目录。	否	true
覆盖文件	bool	是否覆盖文件，当目标路径已存在同名源文件路径时，若 Overwrite 为 false 则抛错，若 Overwrite 为 true 则覆盖写	否	false
重命名	string	移动文件的新名称，如果未提供，则保留原始文件名。当目标路径已存在同名源文件路径时，可设置 RenameTo 参数命名新名称解决移动文件同名冲突，支持表达式输入	否	-

SFTP 移动文件

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

源文件路径

string

请输入字符串

f(x)

目标文件路径

string

请输入字符串

f(x)

创建父级目录

bool

true

覆盖文件

bool

false

重命名

string

请输入字符串

f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 移动文件操作若执行成功 payload 为空，若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含“Code”和“Description”元素，“Code”表示错误类型，“Description”表示错误具体信息，例如：

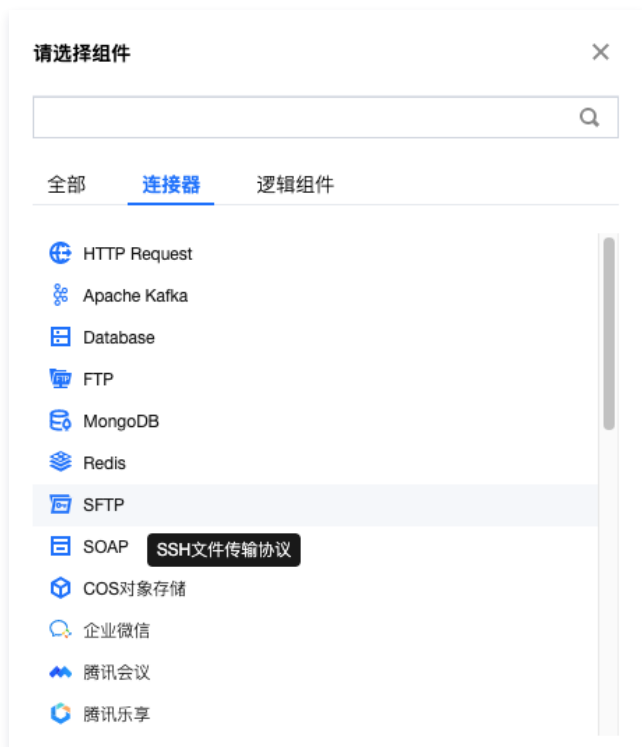
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/sftpctest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 移动文件操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置

选择连接器

配置连接器

通用

连接配置

域名

159.75.18.194

端口号

int

2255

用户名

victorwsw

密码

工作路径

/

SSH配置

SSH公钥认证

bool

false

KNOWN_HOST S密钥验证

bool

false

高级

代理配置

代理域名

请输入字符串

代理端口号

int

请输入整数

代理用户名

请输入字符串

代理密码

请输入字符串

代理协议

请选择

重连接配置

重连接频率(秒)

int

10

超时配置

连接超时时间(秒)

int

60

响应超时时间(秒)

int

60

安全网关

安全网关

无

测试连接

上一步

保存

- 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。源文件路径设为“/data/sftpctest.txt”，目标文件路径设为“/data/movedir”，重命名设为“sftpctestmove.txt”，参数输入见下图：

SFTP ? 移动文件

当前使用的是 sftpbetatestCfg 物理配置
[编辑](#) | [切换](#)

配置 备注

通用

源文件路径 /data/sftpctest.txt f(x)

目标文件路径 /data/movedir f(x)

创建父级目录 true

覆盖文件 false

重命名 sftpctestmove.txt f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

类型选择 +

4. 操作配置参数设置完成后保存返回即可，单击右上角发布，选择发布地域后单击确定。

于 1 分钟前保存
调试
发布
删除
保存

请选择发布地域
☒ 天津
[确定](#) [取消](#)

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 SFTP1 物理配置
[查看](#)

配置 备注

基本配置

监听路径 [复制](#)

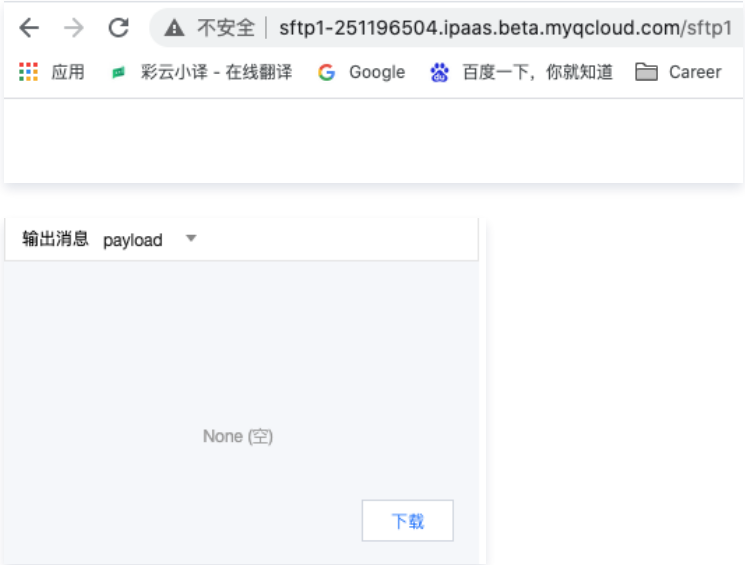
指定监听的路径

[复制监听域名](#)

监听方法

指定监听HTTP请求的方法

- SFTP 移动文件操作若执行成功则返回 payload 响应结果为空。



- 当操作执行失败时，SFTP 输出消息 error 中会设置详细返错信息。



目录遍历

输入参数

参数	数据类型	描述	是否必填	默认值
遍历目录路径	string	要列出的目录的路径支持表达式输入	是	-

子目录递归轮询	bool	是否递归包含子目录的文件和目录	否	false
起始时间	string	在此日期之前创建的文件将被过滤。2020-11-14T08:30:00+00:00支持表达式输入	否	-
截止时间	string	在此日期之后创建的文件将被过滤。2020-11-15T17:30:00+00:00支持表达式输入	否	-
文件名正则匹配	String	使用正则表达式过滤文件或目录名称 [a-z]* 支持表达式输入	否	-
文件路径匹配	String	目录路径过滤匹配模式，只保留指定文件或目录 /mydir/ftpctest.txt，支持表达式输入	否	-
目录	bool	是否过滤目录默认为 true，表示遍历结果中包含目录类型	否	true
文件	bool	是否过滤文件默认为 true，表示遍历结果中包含文件类型	否	true
符号链接	bool	是否过滤符号链接默认为 true，表示遍历结果中包含符号链接类型	否	true
最小比特值	int	文件 Size 过滤最小值	否	-
最大比特值	int	文件 Size 过滤最大值	否	-
文件校验时间间隔(秒)	int	文件大小校验等待时间（秒），以判断文件是否可读。在等待时间前后执行两次 sizecheck 文件大小检测。如果两次检测都返回相同的值，即 sizecheck 期间文件没有其他修改操作则可以读取该文件。（范围：0~60）	否	0 (s)
时间戳使能	bool	判断是否支持根据文件的修改或创建时间戳来判断触发流，缓存持久化记录上次轮询截止时间戳作为下次轮询起始判断时间。	否	false
时间戳状态标识	String	当时间戳使能为 true 时，需设置非空值作为当前流 SFTP 组件时间戳状态缓存持久化的唯一识别标志。	否	-

SFTP 目录遍历

当前无可用连接器配置，需新建连接器配置以便后续调用

新建

配置

备注

通用

遍历目录路径

string 请输入字符串 f(x)

子目录递归轮询

bool false

文件匹配配置

起始时间

string 2000-01-01 00:00:00 f(x)

截止时间

string 2000-01-01 00:00:00 f(x)

文件名正则匹配

string 请输入字符串 f(x)

文件路径匹配

string 请输入字符串 f(x)

目录

bool true

文件

bool true

符号链接

bool true

最小比特值

int 0

最大比特值

int 0

高级

文件校验时间间隔(秒)

int 0

时间戳使能

bool true

时间戳状态标识

string 请输入字符串 f(x)

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 目录遍历操作会在消息 payload 中返回每个文件的属性信息：

参数	数据类型	描述
Name	string	文件名称
Type	string	文件类型：File Folder SymLink
Target	string	SymLink 对应路径
Size	int	文件大小
Time	string	文件最后修改时间
PathName	string	文件绝对路径

输出消息 payload ▾

```
{
  "mytestdir": {
    "PathName": "/data/mytestdir",
    "Size": 4096,
    "Target": null,
    "Time": "2021-03-22 02:13:55",
    "Type": "Folder"
  },
  "sftptest.txt": {
    "PathName": "/data/sftptest.txt",
    "Size": 21,
    "Target": null,
    "Time": "2021-03-22 17:36:04",
    "Type": "File"
  }
}
```

下载

SFTP 目录遍历操作若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含 “Code” 和 “Description” 元素，“Code” 表示错误类型，“Description” 表示错误具体信息，例如：

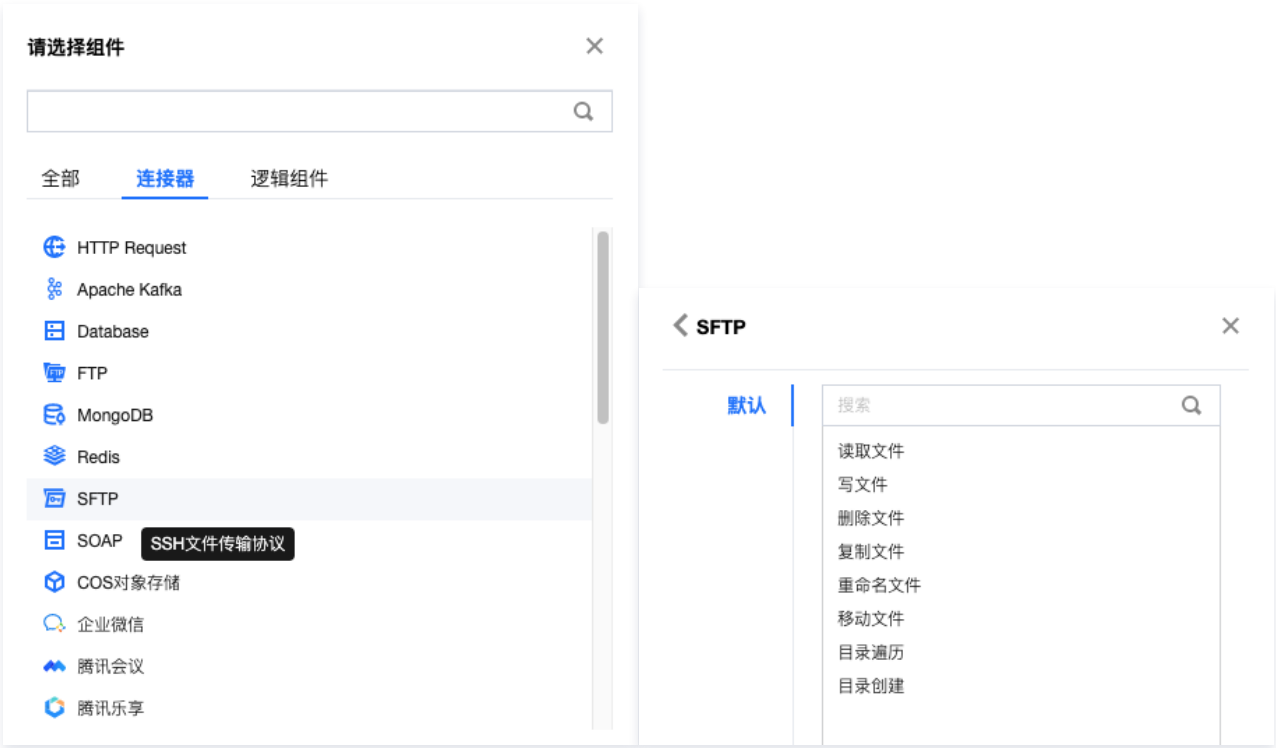
输出消息 error ▾

```
{
  "Code": "CORE:RUNTIME",
  "Description": "SFTP组件配置的directoryPath属性格式错误. directoryPath: @@/data, error: filePathPattern is not matched!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 目录遍历操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的 SFTP 服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方[测试连接](#)校验连接配置参数有效性。具体如下图：

编辑连接器配置



1 选择连接器 > 2 配置连接器

通用

连接配置

域名 *

端口号

用户名

密码

工作路径

SSH配置

SSH公钥认证

KNOWN_HOST S密钥验证

高级

代理配置

代理域名

代理端口号

代理用户名

代理密码

代理协议

重连接配置

重连接频率(秒)

超时配置

连接超时时间 (秒)

响应超时时间 (秒)

安全网关

安全网关

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。遍历目录路径设为“/data”，其他参数输入见下图：

SFTP 目录遍历

当前使用的是 sftpbetatestCfg 物理配置

编辑 | 切换

配置

备注

通用

遍历目录路径

string

/data

f(x)

子目录递归轮询

bool

false

文件匹配配置

起始时间

string

2000-01-01 00:00:00

f(x)

截止时间

string

2000-01-01 00:00:00

f(x)

文件名正则匹配

string

请输入字符串

f(x)

文件路径匹配

string

请输入字符串

f(x)

目录

bool

true

文件

bool

true

符号链接

bool

true

最小比特值

int

0

最大比特值

int

0

高级

文件校验时间间隔(秒)

int

0

时间戳使能

bool

false

4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

于 1 分钟前保存

调试

发布

删除

保存

请选择发布地域

天津

确定 取消

5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

Http Listener ?

当前使用的是 SFTP1 物理配置 [查看](#)

配置

备注

基本配置

监听路径 *

/sftp1

指定监听的路径

复制监听域名

监听方法 *

全部

指定监听HTTP请求的方法

○ SFTP 返回 payload 响应结果如下，即为“/data”目录下遍历文件的属性列表。

```
{
  "copydir": {
    "PathName": "\\.data/copydir",
    "Size": 4096,
    "Target": null,
    "Time": "2021-03-17 14:31:49",
    "Type": "Folder"
  },
  "movedir": {
    "PathName": "\\.data/movedir",
    "Size": 4096,
    "Target": null,
    "Time": "2021-03-17 14:32:25",
    "Type": "Folder"
  },
  "mytestdir": {
    "PathName": "\\.data/mytestdir",
    "Size": 4096,
    "Target": null,
    "Time": "2021-03-22 02:13:55",
    "Type": "Folder"
  },
  "sftpptest.txt": {
    "PathName": "\\.data/sftpptest.txt",
    "Size": 21,
    "Target": null,
    "Time": "2021-03-22 17:36:04",
    "Type": "File"
  }
}
```

输出消息 payload

```
{
  "mytestdir": {
    "PathName": "\\.data/mytestdir",
    "Size": 4096,
    "Target": null,
    "Time": "2021-03-22 02:13:55",
    "Type": "Folder"
  },
  "sftpptest.txt": {
    "PathName": "\\.data/sftpptest.txt",
    "Size": 21,
    "Target": null,
    "Time": "2021-03-22 17:36:04",
    "Type": "File"
  }
}
```

下载

○ 当操作执行失败时，SFTP 输出消息 error 中会设置详细返错信息。



500

CORE:RUNTIME

SFTP组件配置的directoryPath属性格式错误. directoryPath: @@/data, error: filePathPattern is not matched! (TraceId: 1b7b835d-d6a5-e6a8-d297-d1298e0d4f56)

输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "SFTP组件配置的directoryPath属性格式错误. directoryPath: @@/data, error: filePathPattern is not matched!"
}
```

下载

目录创建

输入参数

参数	数据类型	描述	是否必填	默认值
目录创建路径	string	新创建目录的名称默认支持创建多级目录，当路径已存在时返错支持表达式输入	是	—

SFTP 目录创建

当前使用的是 sftpbetatestCfg 物理配置

配置

备注

通用

目录创建路径

string 请输入字符串 f(x)

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

输出

组件输出的 message 信息如下：

message 属性	值
payload	执行成功或失败，payload 均为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

SFTP 目录创建操作若执行成功 payload 为空，若执行失败时输出消息 error 中会设置详细返错信息。错误输出为 dict 类型，包含 “Code” 和 “Description” 元素，“Code” 表示错误类型，“Description” 表示错误具体信息，例如：

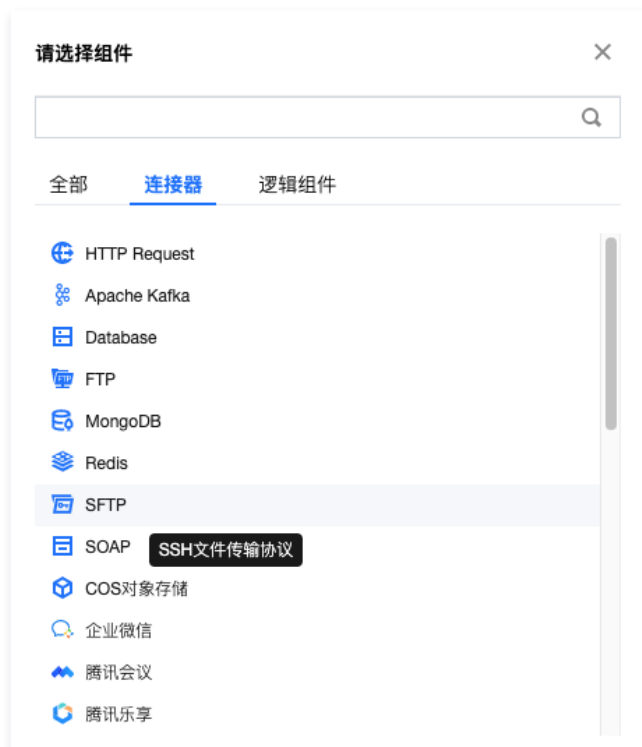
输出消息 error

```
{
  "Code": "CORE:RUNTIME",
  "Description": "path ./data/sftptest1.txt not exists!"
}
```

下载

案例

1. 在连接器列表中选择 SFTP 连接器，然后选择 SFTP 目录创建操作。



2. 输入连接器配置参数，其中域名、端口、用户名和密码设置为待连接的SFTP服务器对应参数，工作路径设置为 SFTP 服务器下文件处理的相对路径，默认为根路径 “/”，其他参数保持默认值即可。参数填写完后可单击下方测试连接校验连接配置参数有效性。具体如下图：

编辑连接器配置



1 选择连接器 > 2 配置连接器

通用

连接配置

域名 *

端口号

用户名

密码 

工作路径

SSH配置

SSH公钥认证

KNOWN_HOST S密钥验证

高级

代理配置

代理域名

代理端口号

代理用户名

代理密码

代理协议

重连接配置

重连接频率(秒)

超时配置

连接超时时间(秒)

响应超时时间(秒)

安全网关

安全网关

测试连接

上一步

保存

3. 确认连接器配置参数填写无误后单击**保存**，然后设置操作配置参数。文件路径设为“/data/a/b/c”，参数输入见下图：



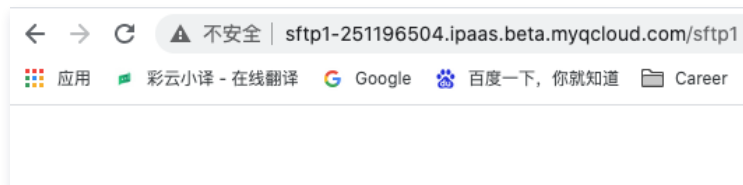
4. 操作配置参数设置完成后保存返回即可，然后单击右上角**发布**，选择发布地域后单击**确定**。

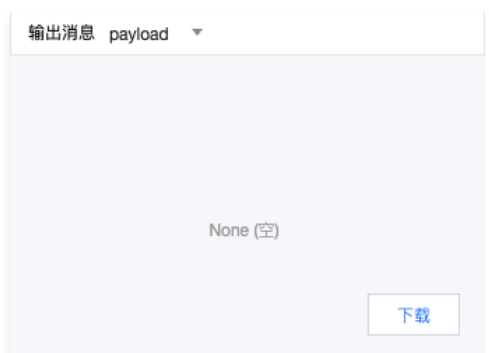


5. 待集成流发布成功后复制 HTTP Listener 监听路径后访问该域名即可触发流。

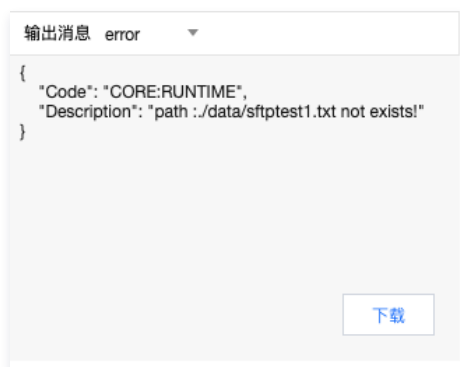
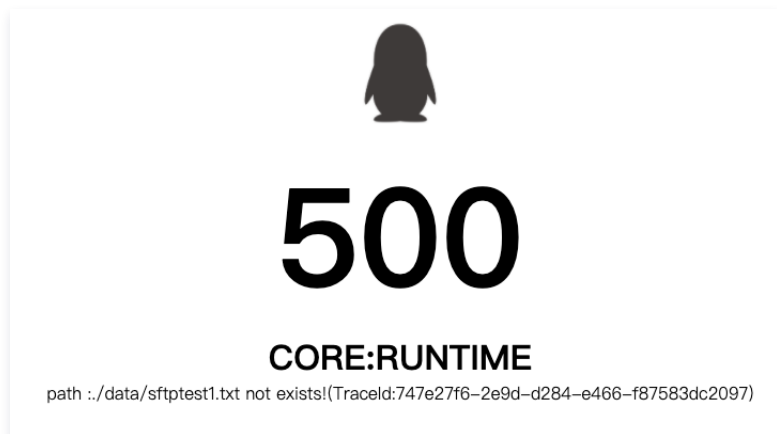


○ SFTP 目录创建操作若执行成功则返回 payload 响应结果为空。





- 当操作执行失败时，SFTP 输出消息 error 中会设置详细返错信息。



Email 连接器

最近更新时间：2023-06-08 14:48:02

简介

Email 连接器可用于发送和收取邮件。

- 邮件发送 SMTP 协议
SMTP 是一个相对简单的基于文本的协议，在其之上指定一条消息的一个或多个接收者（在大多数情况下被确认是存在的），然后进行消息文本传输。SMTP 规定在两个相互通信的 SMTP 进程之间应如何交换信息。由于 SMTP 使用 CS 方式，因此负责发送邮件的 SMTP 进程为 SMTP 客户，而负责接收邮件的 SMTP 进程为 SMTP 服务器。至于邮件内部的格式，邮件如何存储，以及邮件系统应以多快的速度来发送邮件，SMTP 未做出规定。
- 邮件读取 POP3 和 IMAP 协议
常用的邮件读取协议有两个，即邮局协议第3个版本 POP3 和网际报文存取协议 IMAP (Internet Message Access Protocol)。
- 邮局协议（英语：Post Office Protocol，缩写：POP）是 TCP/IP 协议族中的一员，由1996年5月发行之 RFC 1939 首次定义。此协议主要用于支持使用客户端远程管理在服务器上的 电子邮件。最新版本为 POP3，全名“Post Office Protocol – Version 3”，而提供 SSL 加密的 POP3 协议则被称为 POP3S。
- 因特网信息访问协议（英语：Internet Message Access Protocol，缩写：IMAP；以前称作交互邮件访问协议）是一个应用层协议，用来从本地邮件客户端（例如：Microsoft Outlook、Foxmail、Mozilla Thunderbird）访问远程服务器上的邮件。在使用 IMAP 时，在用户的计算机上运行 IMAP 客户程序，然后与接收方的邮件服务器上的 IMAP 服务器程序建立 TCP 连接。用户在计算机上即可操纵邮件服务器的邮箱，就像在本地操纵一样，因此 IMAP 是一个联机协议。

连接器配置

Email 连接器需提供如下连接配置，POP3 协议和 IMAP 协议的连接信息可只提供其中一种，或同时提供。

Email 连接器配置参数如下：

基础设置				
参数	数据类型	描述	是否必填	默认值
默认发件人地址	string	默认发件人地址	是	—
用户名	string	邮箱登录用户名	是	—
密码	string	邮箱登录密码，163邮箱、QQ 邮箱等邮箱使用的是授权码，需要开通 POP3 或 IMAP 收件服务后，使用授权码登录	是	—
协议类型	enum	选择收信的协议类型：可选 POP3 或 IMAP 或 ALL	否	ALL
超时时间	int	设置连接的超时时间	否	10s

POP 设置				
参数	数据类型	描述	是否必填	默认值
POP3 服务器地址	string	POP3 服务器地址	是	—
POP3 服务器端口号	int	POP3 服务器端口号	是	—
POP3 使能 TLS	bool	是否使用 TLS 链接	否	false
POP3 TLS 连接客户端证书	file	可选，使用提供的证书对连接进行加密	否	—
POP3 TLS 连接客户端 Key	file	可选，使用提供的证书对连接进行加密，需和客户端证书同时提供	否	—
POP3 TLS 连接服务端证书	file	可选，使用提供的证书对连接进行加密	否	—

IMAP 设置

参数	数据类型	描述	是否必填	默认值
IMAP 服务器地址	string	POP3 服务器地址	是	–
IMAP 服务器端口号	int	POP3 服务器端口号	是	–
IMAP 使能 TLS	bool	是否使用 TLS 链接	否	false
IMAP TLS 连接客户端证书	file	可选，使用提供的证书对连接进行加密	否	–
IMAP TLS 连接客户端 Key	file	可选，使用提供的证书对连接进行加密，需和客户端证书同时提供	否	–
IMAP TLS 连接服务端证书	file	可选，使用提供的证书对连接进行加密	否	–

SMTP 设置

参数	数据类型	描述	是否必填	默认值
SMTP 服务器地址	string	POP3 服务器地址	是	–
SMTP 服务器端口号	int	POP3 服务器端口号	是	–
SMTP 使能 TLS	bool	是否使用 TLS 链接	否	false
SMTP TLS 连接客户端证书	file	可选，使用提供的证书对连接进行加密	否	–
SMTP TLS 连接客户端 Key	file	可选，使用提供的证书对连接进行加密，需和客户端证书同时提供	否	–
SMTP TLS 连接服务端证书	file	可选，使用提供的证书对连接进行加密	否	–

连接器配置界面如下：

编辑连接器配置



- ✓ 选择连接器
- >
- 2 配置连接器

基础设置

默认发件人地址 *

string

xxxxxx@163.com

f(x)

用户名 *

fxxxxxx@163.com

密码 *

.....

账户密码

协议类型

ALL

超时时间(秒)

int

10

POP设置

服务器名称 *

pop.163.com

端口号 *

int

995

使能TLS

bool

true

客户端证书文件内容

还未选择文件

选择文件

客户端key文件内容

还未选择文件

选择文件

服务端证书文件内容

还未选择文件

选择文件

IMAP设置

操作说明

Email 连接器包含发送邮件、删除邮件、标记邮件、清理文件夹、收取邮件-IMAP、收取邮件-POP3 共6种操作。

发送邮件

输入参数

参数	数据类型	说明	是否必填	默认值
发件人	string	发件人地址	否	–
收件人	string	收件人地址列表，至少提供一个收件地址	是	–
抄送	string	抄送地址列表	否	–
密送	string	密送地址列表	否	–
回复地址	string	回复地址列表	否	–
主题	string	邮件主题	否	无主题
邮件头	map	邮件头	否	–
正文	string	邮件正文	否	–
正文MIME类型	枚举	正文 MIME 类型，可选 text/plain、text/html	否	text/plain
附件	Attachment	附件列表	否	–
编码格式	枚举	编码格式	否	base64

Attachment 参数

参数	数据类型	说明	是否必填	默认值
文件名	string	附件文件名	是	–
文件内容	byte	文件内容，仅支持使用 dataaway 表达式	是	–

Email ? 发送邮件

...

☐

当前使用的是 email 物理配置

编辑 | 切换

配置

备注

通用 ▲

发件人

string ▼

请输入字符串

f(x)

收件人 *

☒ 列表添加

☐ f(x)

value

操作

item_1

删除

+添加

收件人地址

抄送

☒ 列表添加

☐ f(x)

value

操作

+添加

密送

☒ 列表添加

☐ f(x)

value

操作

+添加

回复地址

☒ 列表添加

☐ f(x)

value

操作

+添加

主题

string ▼

无主题

f(x)

邮件头

☒ 列表添加

☐ f(x)

输出参数

发送邮件操作执行成功后，继承上个组件的 payload 信息；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息

error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值不变。执行失败后，message error 值（error 值不固定）如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in http://service.mail.qq.com/cgi-bin/help?subtype=1&id=28&no=1001256\"
}
```

案例

- 1. 在邮箱设置中打开 POP3 或 IMAP 协议，例如：在163邮箱中打开 POP3 和 IMAP 协议，并记住授权码。
 - 在 Outlook 邮箱设置中，找到“同步电子邮件”选项卡，并打开 POP。Outlook 邮箱打开 POP 的设置方式如下图：



- QQ 邮箱、163邮箱可参考以下帮助文档：
 - QQ 邮箱：[QQ 邮箱授权码设置](#)、[QQ 邮箱开启 POP3/SMTP/IMAP 功能](#)。
 - 163邮箱：[163邮箱设置](#)。
 - 其他邮箱可在邮箱帮助文档中查看 POP3 及 IMAP 相关主题。
- 2. 按照邮箱帮助设置 IMAP、POP3、SMTP 服务器地址和端口号，在连接设置中填写正确的地址信息，填写用户名（xxx@163.com）和密码（163、QQ 等邮箱使用的授权码）；使用 ssl 的端口时，使能 TLS 开关需设置为 True。
- 3. 单击测试连接，查看设置是否正确。
- 4. 填写邮件的收件人、主题、正文等信息；附件信息必须使用 dataway，仅接受字节数组作为附件的内容。
 - 添加附件示例：

在附件栏单击添加。

Email 发送邮件

当前使用的是 email 物理配置

编辑 | 切换

配置	备注
回复地址	列表添加 f(x) value 操作 +添加
主题	string 无主题 f(x)
邮件头	列表添加 f(x) value 操作 +添加
正文	string 请输入字符串 f(x)
正文MIME类型	text/plain f(x)
附件	列表添加 f(x) value 操作 +添加
编码格式	base64 f(x)
高级配置	
输出消息绑定自定义数据类型	
消息属性	payload

依次输入附件文件名、附件内容。

注意

附件内容必须是字节数组。

item_1



☒ 手动输入 ☐ f(x)

附件名称 *

string ▼

test.txt

f(x)

附件内容，必
须是byte数组 *

def dw_process(msg): return



确定

取消

删除邮件

删除邮件将指定文件夹中的指定 ID 的邮件标记为删除。

输入参数

参数	数据类型	说明	是否必填	默认值
邮件文件夹	string	操作的邮件文件夹	否	INBOX
邮件ID	int	要删除的邮件 ID	是	-

输出参数

删除操作执行成功后，继承上个组件的 payload 信息；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值不变。执行失败后，message error 值（error 值不固定）如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in http://service.mail.qq.com/cgi-bin/help?subtype=1&id=28&no=1001256\"
}
```

案例

1. 使用发送操作中的邮件连接信息。
2. 填写邮件文件夹，并填写邮件 ID（在收取邮件操作中，会返回邮件id及其内容），例如：下图的配置将 INBOX 文件夹（收件夹）下指定邮件 ID 的邮件标记为删除。

 Email  删除邮件




当前使用的是 email 物理配置

[编辑](#) | [切换](#)

配置

备注

通用

邮件文件夹

string ▼

INBOX

f(x)

邮件ID *

int ▼

1

f(x)

清理文件夹

清理文件夹将删除文件夹中标记为删除的邮件。

输入参数

参数	数据类型	说明	是否必填	默认值
邮件文件夹	string	操作的邮件文件夹	否	INBOX

输出参数

清理文件夹操作执行成功后，继承上个组件的 payload 信息；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值不变。执行失败后，message error 值（error 值不固定）如下：

```

{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in http://service.mail.qq.com/cgi-bin/help?subtype=1&id=28&no=1001256\"
}
```


- 案例
1. 使用发送操作中的邮件连接信息。

2. 填写邮件文件夹，下图的配置将删除掉 INBOX 文件夹下的标记为删除的全部邮件（注意：此功能受邮件服务器设置影响，如设置为不允许客户端删信，则此动作不会生效）。

Email ? 清理文件夹

当前使用的是 email 物理配置 编辑 | 切换

配置 备注

通用

邮件文件夹 string INBOX f(x)

标记邮件

标记邮件将指定文件夹中的指定 ID 的邮件标记为删除、已读或旗标。

输入参数

参数	数据类型	说明	是否必填	默认值
邮件文件夹	string	操作的邮件文件夹	否	INBOX
邮件 ID	int	要删除的邮件 ID	是	-
标记为	枚举	将指定 ID 的邮件标记为：删除、已读、旗标	是	-

输出参数

标记邮件操作执行成功后，继承上个组件的 payload 信息；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	继承上个组件的 payload 信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值不变。执行失败后，message error 值（error 值不固定）如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in http://service.mail.qq.com/cgi-bin/help?subtype=1&id=28&no=1001256\"
```

```
}

```

案例

- 1. 使用发送操作中的邮件连接信息。
- 2. 填写邮件文件夹，并填写邮件 ID（在收取邮件中，会返回邮件 ID 及其内容）；
- 3. 选择标记动作。下图的配置，将 INBOX 文件夹（收件夹）中邮件 ID 为1的邮件标记为已读。

 Email  标记邮件 ... 

当前使用的是 email 物理配置 编辑 | 切换

配置备注

通用

邮件文件夹

string

INBOX

f(x)

邮件ID *

int

1

f(x)

标记为 *

已读

f(x)

收取邮件（POP3 协议）

收取邮件（POP3 协议）按照指定的过滤条件，收取符合过滤条件的全部邮件（不能指定邮件文件夹）。

输入参数

参数	数据类型	说明	是否必填	默认值
邮件过滤条件	POP3Matcher	邮件过滤条件	否	-
最大收取量	int	最大收取量	否	10
起始邮件 ID	int	起始邮件 ID	否	0

POP3Matcher

参数	数据类型	说明	是否必填	默认值
起始时间	datetime	收取此时间之后的邮件	否	-
截止时间	datetime	收取此时间之前的邮件	否	-
主题筛选	string	筛选邮件主题，收取主题中包含该字段的邮件	否	-
发件人地址筛选	string	筛选邮件发件人，收取发件人地址中包含该字段的邮件	否	-

收件人地址筛选	string	筛选邮件收件人，收取收件人地址中包含该字段的邮件	否	-
---------	--------	--------------------------	---	---

输出参数

收取邮件（POP3 协议）操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，键表示邮件各字段名称，值表示邮件字段值
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  {
    "Header":Object{...},
    "Subject":" some subject",
    "Sender":null,
    "From":[
      {
        "Name":"帐户名称",
        "Address":"abc@efg.com"
      }
    ],
    "ReplyTo":null,
    "To":[
      {
        "Name":"","
        "Address":"receiverAddress@qq.com"
      }
    ],
    "Cc":null,
    "Bcc":null,
    "Date":"2021-03-01T23:42:53-08:00",
    "MessageID":"IKBLXSIQ3DU4.PKYL66BZ4RFP3@BY1PEPF00001EF1",
    "MailID":1,
    "InReplyTo":null,
    "References":null,
    "ResentFrom":null,
    "ResentSender":null,
    "ResentTo":null,
    "ResentDate":"0001-01-01T00:00:00Z",
    "ResentCc":null,
    "ResentBcc":null,
    "ResentMessageID":"","
    "ContentType":"","
    "Content":"","
    "HTMLBody":"... HTML Body Content ...",
    "TextBody":"... Text Body Content ...",
    "Attachments":null,
    "EmbeddedFiles":null
  },
  {
    "Header":{},
    "Subject":"邮件主题",
    "Sender":null,
```

```
"From": [
  {
    "Name": "帐户名称",
    "Address": "abc@efg.com"
  }
],
"ReplyTo": null,
"To": [
  {
    "Name": "",
    "Address": "receiverAddress@qq.com"
  }
],
"Cc": null,
"Bcc": null,
>Date": "2021-03-08T00:05:34-08:00",
"MessageID": "GM63WJUK5DU4.Y7UK7WAJ3VBS2@BL02EPF0000195A",
"MailID": 2,
"InReplyTo": null,
"References": null,
"ResentFrom": null,
"ResentSender": null,
"ResentTo": null,
"ResentDate": "0001-01-01T00:00:00Z",
"ResentCc": null,
"ResentBcc": null,
"ResentMessageID": "",
"ContentType": "",
"Content": "",
"HTMLBody": "",
"TextBody": "",
"Attachments": [
  {
    "FileName": "test.pdf",
    "ContentType": "application/pdf; name=test.pdf",
    "Data": "base64 encoded content"
  },
  ...
]
},
"EmbeddedFiles": [
  {
    "CID": "colors",
    "ContentType": "image/png; name=colors.png",
    "Data": "base64 encoded content"
  },
  ...
]
}
```

执行失败后，message error 值（error 值不固定）如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in
http://service.mail.qq.com/cgi-bin/help?subtype=1&id=28&no=1001256\"
}
```

案例

1. 使用发送操作中的邮件连接信息。
2. 填写过滤条件，参考输入条件中的 POP3Matcher；填写最大收取量和起始邮件 ID。按下图示例填写的配置，将收取邮件服务器收信时间在2021-01-01 10:28:38 ~ 2021-04-25 10:28:38之间、邮件主题包含 test、发件人包含 someone、收件人包含 somereceiver 的邮件，邮件 ID 从1开始，共收取10封邮件（若满足条件的邮件数不足10，则返回全部满足条件的邮件）。

Email ? 收取邮件(P...

当前使用的是 email 物理配置

编辑 | 切换

配置

备注

通用

邮件过滤条件

☒ 手动控制
 ☐ f(x)

起始时间

datetime

2021-01-01 10:28:38

📅

f(x)

收取这个时间之后的邮件

截止时间

datetime

2021-04-25 10:28:38

📅

f(x)

收取这个时间之前的邮件

主题筛选

string

test

f(x)

按主题正则匹配

发件人地址筛选

string

someone

f(x)

按发件人地址正则匹配

收件人地址筛选

string

somereceiver

f(x)

按收件人地址正则匹配

最大收取量

int

10

f(x)

收取邮件时一次收取的最大数量

起始邮件ID

int

0

f(x)

起始邮件ID，将只收取该ID之后的邮件

高级配置

输出消息绑定自定义数据类型

收取邮件（IMAP 协议）

收取邮件（IMAP 协议）按照指定的过滤条件，收取符合过滤条件的全部邮件。

输入参数

参数	数据类型	说明	是否必填	默认值
邮件文件夹	string	操作的邮件文件夹名称	否	INBOX
邮件过滤条件	IMAPMatcher	邮件过滤条件	否	-
最大收取量	int	最大收取量	否	10
起始邮件 ID	int	起始邮件 ID	否	0

IMAPMatcher

参数	数据类型	说明	是否必填	默认值
已读	bool	取值为 False 时，仅收取未读邮件；取值为 True 时，仅收取已读邮件；取值为 None 时，不使用该条件过滤	否	None
已回复	bool	取值为 False 时，仅收取未回复邮件；取值为 True 时，仅收取已回复邮件；取值为 None 时，不使用该条件过滤	否	None
旗标	bool	取值为 False 时，仅收取无旗标邮件；取值为 True 时，仅收取旗标邮件；取值为 None 时，不使用该条件过滤	否	None
起始时间	datetime	收取此时间之后的邮件	否	-
截止时间	datetime	收取此时间之前的邮件	否	-
主题筛选	string	筛选邮件主题，收取主题中包含该字段的邮件	否	-
发件人地址筛选	string	筛选邮件发件人，收取发件人地址中包含该字段的邮件	否	-
收件人地址筛选	string	筛选邮件收件人，收取收件人地址中包含该字段的邮件	否	-

输出参数

收取邮件（IMAP 协议）操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，键表示邮件各字段名称，值表示邮件字段值
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
[
  {
    "Header":Object{...},
    "Subject": " some subject",
    "Sender":null,
    "From":[
      {
        "Name": "帐户名称",
        "Address": "abc@efg.com"
      }
    ],
    "ReplyTo":null,
    "To":[
      {
        "Name": "",
        "Address": "receiverAddress@qq.com"
      }
    ],
    "Cc":null,
    "Bcc":null,
```

```
"Date": "2021-03-01T23:42:53-08:00",
"MessageID": "IKBLXSIQ3DU4.PKYL66BZ4RFP3@BY1PEPF00001EF1",
"MailID": 1,
"InReplyTo": null,
"References": null,
"ResentFrom": null,
"ResentSender": null,
"ResentTo": null,
"ResentDate": "0001-01-01T00:00:00Z",
"ResentCc": null,
"ResentBcc": null,
"ResentMessageID": "",
"ContentType": "",
"Content": "",
"HTMLBody": "... HTML Body Content ...",
"TextBody": "... Text Body Content ...",
"Attachments": null,
"EmbeddedFiles": null
},
{
  "Header": {},
  "Subject": "邮件主题",
  "Sender": null,
  "From": [
    {
      "Name": "帐户名称",
      "Address": "abc@efg.com"
    }
  ],
  "ReplyTo": null,
  "To": [
    {
      "Name": "",
      "Address": "receiverAddress@qq.com"
    }
  ],
  "Cc": null,
  "Bcc": null,
  "Date": "2021-03-08T00:05:34-08:00",
  "MessageID": "GM63WJUK5DU4.Y7UK7WAJ3VBS2@BL02EPF0000195A",
  "MailID": 2,
  "InReplyTo": null,
  "References": null,
  "ResentFrom": null,
  "ResentSender": null,
  "ResentTo": null,
  "ResentDate": "0001-01-01T00:00:00Z",
  "ResentCc": null,
  "ResentBcc": null,
  "ResentMessageID": "",
  "ContentType": "",
  "Content": "",
  "HTMLBody": "",
  "TextBody": "",
  "Attachments": [
    {
      "FileName": "test.pdf",
      "ContentType": "application/pdf; name=test.pdf",
      "Data": "base64 encoded content"
    },
    ...
  ]
}
```

```
    ],
    "EmbeddedFiles": [
      {
        "CID": "colors",
        "ContentType": "image/png; name=colors.png",
        "Data": "base64 encoded content"
      },
      ...
    ]
  }
]
```

执行失败后，message error 值（error 值不固定）如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "535 Login Fail. Please enter your authorization code to login. More information in
http://service.mail.qq.com/cgi-bin/help?subtype=1&&id=28&&no=1001256\"
}
```

案例

1. 使用发送操作中的邮件连接信息。
2. 填写邮件文件夹、过滤条件（参考输入条件中的 IMAPMatcher）、填写最大收取量和起始邮件 ID。按下图示例填写的配置，将收取邮件服务器收信时间在 2021-01-01 10:28:38 ~ 2021-04-25 10:28:38 之间、邮件主题包含 test、发件人包含 someone、收件人包含 somereceiver 的邮件，邮件 ID

从1开始，且是已读的邮件，共收取10封邮件（若满足条件的邮件数不足10，则返回全部满足条件的邮件）。

Email ? 收取邮件(l... ..

当前使用的是 email 物理配置

编辑 | 切换

配置

备注

邮件文件夹

string INBOX f(x)

操作的邮件文件夹名称

邮件过滤条件

☒ 手动控制
 ☐ f(x)

已读?

True f(x)

收取已读邮件(True)/未读邮件(False)/不选择(None)

已回复?

None f(x)

收取已回复邮件(True)/未回复邮件(False)/不选择(None)

旗标?

None f(x)

收取旗标邮件(True)/无旗标邮件(False)/不选择(None)

起始时间

datetim 2021-01-01 10:28:38 f(x)

收取这个时间之后的邮件

截止时间

datetim 2021-04-25 10:28:38 f(x)

收取这个时间之前的邮件

主题筛选

string test f(x)

按主题正则匹配

发件人地址筛选

string someone f(x)

按发件人地址正则匹配

收件人地址筛选

string somereceiver f(x)

按收件人地址正则匹配

最大收取量

int 10 f(x)

收取邮件时一次收取的最大数量

起始邮件ID

int 0 f(x)

起始邮件ID，将只收取该ID之后的邮件

状态存储连接器

最近更新时间：2024-10-18 16:22:22

简介

状态存储以抽象数据结构的方式为用户提供多种类型的持久化存储方式，主要有以下特点：

- 提供字符串（string）、列表（list）、哈希（hash）等数据结构的存储。
- 支持数据的持久化，数据保持在磁盘中，流重启的时候可以再次加载进行使用。
- 支持的数据类型：
 - 字符串（string）：字符串，字符串与键一一对应。
 - 列表（list）：简单字符串列表，按照插入顺序排序，列表与键一一对应。
 - 哈希（hash）：字符串类型的key和value的映射表，映射表与键一一对应。

连接器配置

配置参数

参数	数据类型	描述	是否必填
键	string	状态的唯一标识，长度限制30字符	是

配置界面

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键

string

请输入字符串

f(x)

上一步

保存

操作说明

字符串

字符串支持的操作：设置字符串的值、获取字符串的值、删除字符串、设置字符串的过期时间。

设置字符串的值

参数配置

参数	数据类型	描述	是否必填	默认值
值	string	需要更新的值，长度限制4GB	是	-

配置

调试

备注

通用

值

string

value

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示设置后的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "value": "value"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

- 1. 添加状态存储组件，选择设置字符串的值操作。





2. 新建状态存储配置，配置键。



3. 执行操作。设置成功后，message payload 中包含设置后数据：

```
{
  "value": "value"
}
```

获取字符串的值

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示获取的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "value": "value"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择获取字符串的值操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中包含获取的数据。

```
{
  "value": "value"
}
```

删除字符串

参数配置

仅需配置通用配置。

输出

执行成功时，没有结果返回；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on 'localhost'"
}
```

案例

1. 添加状态存储组件，选择删除字符串操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。删除成功后，message payload 为空。

设置字符串的过期时间

参数配置

参数	数据类型	描述	是否必填	默认值
过期时间	int	过期时间（单位天）	是	-

通用

过期时间(天) *

int

1

f(x)

输出

执行成功时，无结果返回；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
```

```
"Description": "Can't connect to MySQL server on 'localhost'"
}
```

案例

1. 添加状态存储组件，选择设置字符串的过期时间操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。设置成功后，message payload 为空。

列表

列表支持的操作：插入值到列表尾部、插入值到列表头部、获取列表的尾部元素、获取列表的头部元素、移出列表尾部元素、移出列表头部元素、获取列表长度、删除列表、设置列表的过期时间。

插入值到列表尾部

参数配置

参数	数据类型	描述	是否必填	默认值
值	string	需要更新的值，长度限制4G	是	

通用

值 *

string

test

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示插入的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{  
  "value": "test"  
}
```

执行失败后，message error 值如下：

```
{  
  "Code": "CORE:RUNTIME",  
  "Description": "Can't connect to MySQL server on '*'"  
}
```

案例

1. 添加状态存储组件，选择插入值到列表尾部操作。

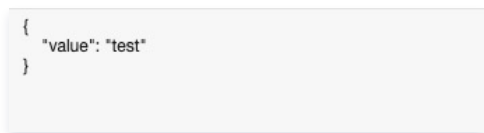




2. 新建状态存储配置，配置键。



3. 执行操作。插入成功后，message payload 中包含插入数据：



插入值到列表头部

参数配置

参数	数据类型	描述	是否必填	默认值
值	string	需要更新的数据，长度限制4GB	是	-

通用

值 *

stringtestf(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示插入的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "value": "test"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

- 1. 添加状态存储组件，选择插入值到列表头部操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。插入成功后，message payload 中包含插入数据：

```
{
  "value": "test"
}
```

获取列表的尾部元素

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示获取的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

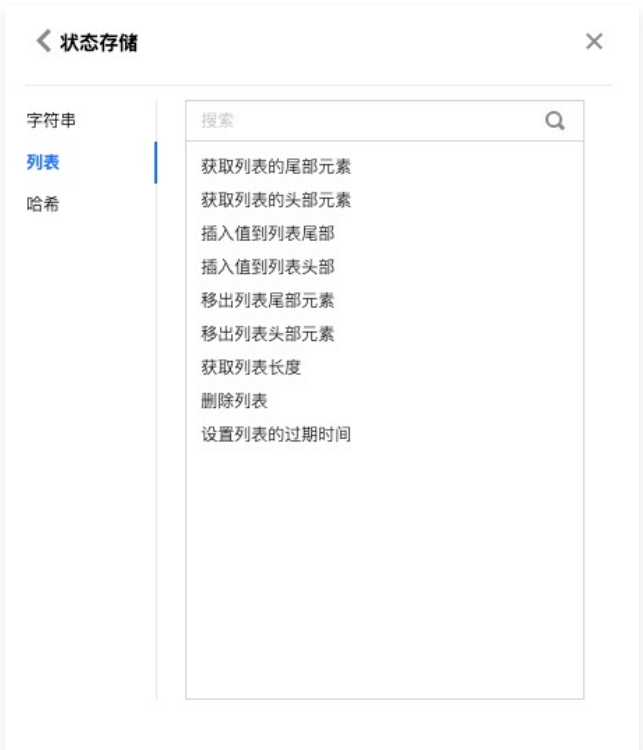
```
{
  "value": "test"
}
```

执行失败后，message error 值如下：


```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择获取列表的尾部元素操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中包含获取的数据：

```
{
  "value": "test"
}
```

获取列表的头部元素

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示获取的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

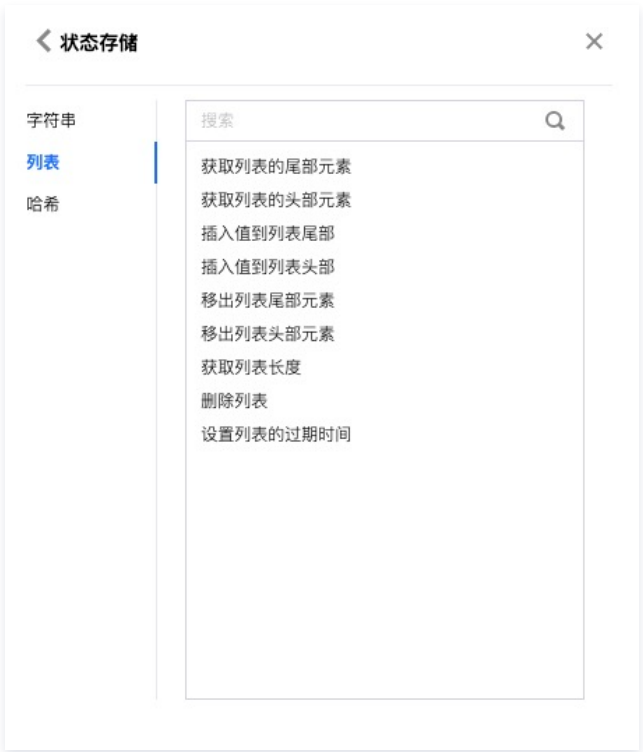
```
{
  "value": "test"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择获取列表的头部元素操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中包含获取的数据：

```
{
  "value": "test"
}
```

移出列表尾部元素

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示被移出的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

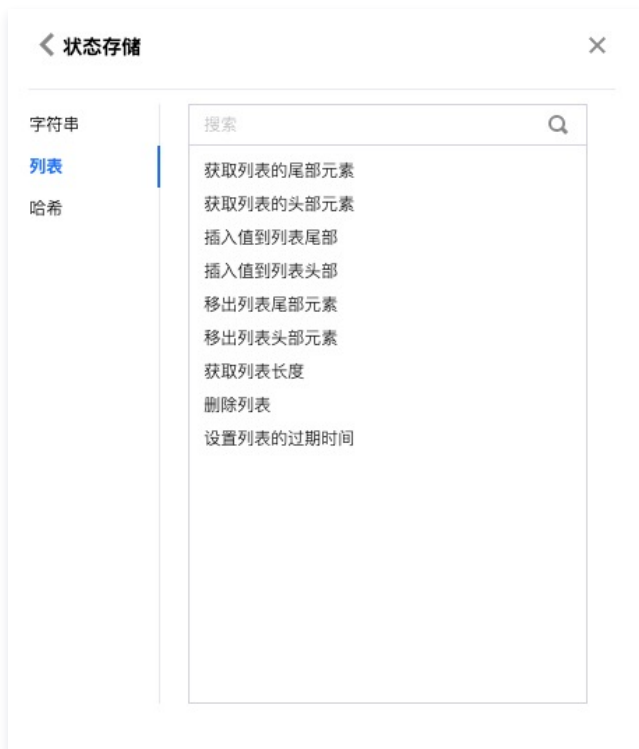
```
{
  "value": "test"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择移出列表尾部元素操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。移出成功后，message payload 中包含被移出的数据：

```
{
  "value": "test"
}
```

移出列表头部元素

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"value"元素，"value"表示移出的数据；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

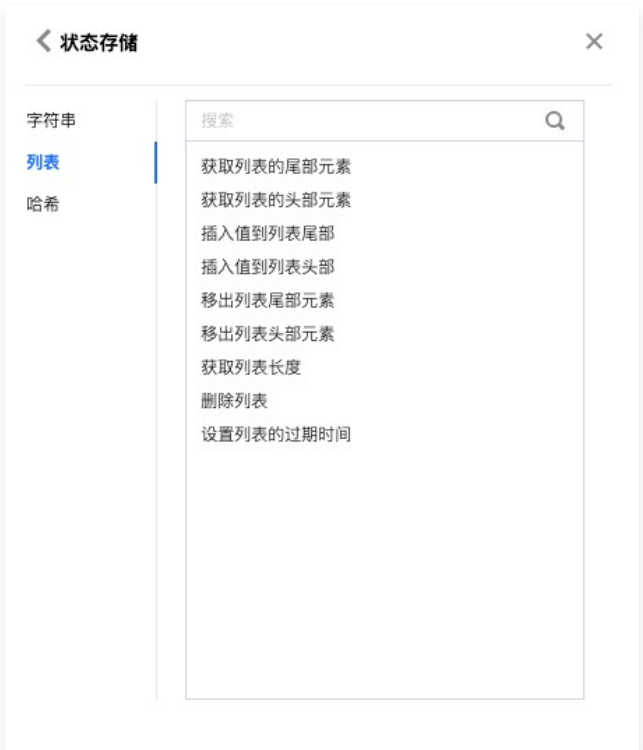
```
{
  "value": "test"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择移出列表头部元素操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。移出成功后，message payload 中包含被移出的数据：

```
{
  "value": "test"
}
```

获取列表长度

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"size"元素，"size"表示获取的长度；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "size": 1
}
```

执行失败后，message error 值如下：


```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择获取列表长度操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中包含获取的长度：

```
{
  "size": 1
}
```

删除列表

参数配置

仅需配置通用配置。

输出

执行成功时，无结果返回；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择删除列表操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。删除成功后，message payload 为空。

设置列表的过期时间

参数配置

参数	数据类型	描述	是否必填	默认值
过期时间	int	过期时间（单位天）	是	-

通用

过期时间(天) *

int

1

f(x)

输出

执行成功时，无结果返回；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
```

```
"Code": "CORE:RUNTIME",
"Description": "Can't connect to MySQL server on '*'"
```

案例

1. 添加状态存储组件，选择设置列表的过期时间操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。设置成功后，message payload 为空。

哈希

哈希支持的操作：设置哈希表中指定字段的值、获取哈希表中指定字段的值、删除哈希表中指定字段、检查哈希表中指定字段是否存在、获取哈希表中字段的数量、删除哈希表、设置哈希表的过期时间。

设置哈希表中指定字段的值

如果字段存在，则会将字段对应的值更新为新值。

参数配置

参数	数据类型	描述	是否必填	默认值
字段	string	需要更新的字段，长度限制512字符	是	-
值	string	需要更新的值，长度限制4GB	是	-

字段 *

string

key

f(x)

值 *

string

test

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"key"和"value"元素，"key"表示设置的字段，"value"表示设置的值；执行失败后，payload 为空

error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "key": "key",
  "value": "test"
}
```

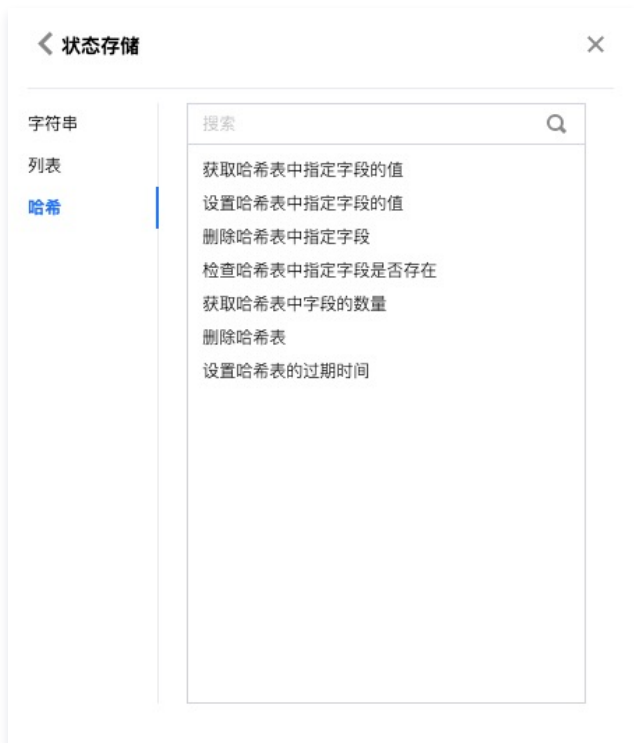
执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

- 1. 添加状态存储组件，选择设置哈希表中指定字段的值操作。





2. 新建状态存储配置，配置键。



3. 执行操作。设置成功后，message payload 中包含设置后的数据：

```
{
  "key": "key",
  "value": "test"
}
```


获取哈希表中指定字段的值

参数配置

参数	数据类型	描述	是否必填	默认值
字段	string	需要获取的字段，长度限制512个字符	是	

通用

字段

string

key

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 Message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"key"和"value"元素，"key"表示获取的字段，"value"表示获取的值；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

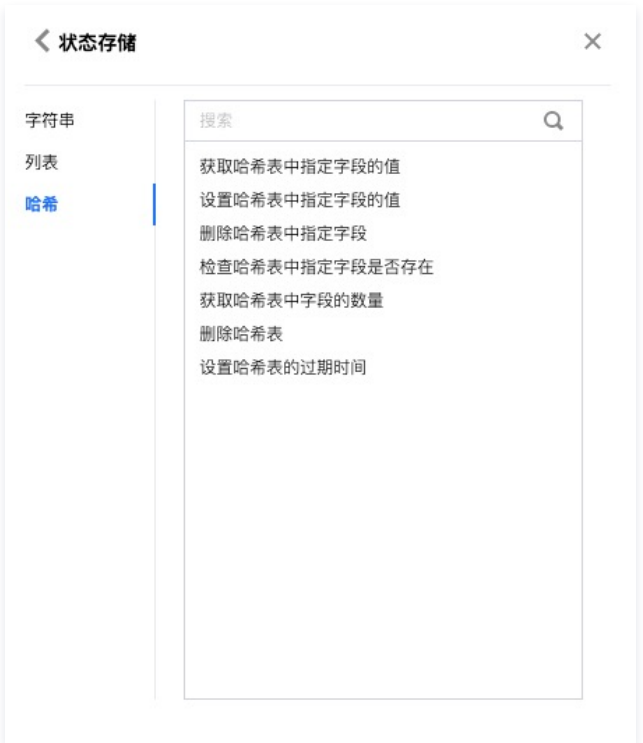
```
{
  "key": "key",
  "value": "test"
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

- 1. 添加状态存储组件，选择获取哈希表中指定字段的值操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message 的 payload 中包含获取的数据：

```
{
  "key": "key",
  "value": "test"
}
```

删除哈希表中指定字段

参数配置

参数	数据类型	描述	是否必填	默认值
字段	string	需要删除字段，长度限制512个字符	是	-

通用

字段 *

string

key

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"key"和"value"元素，"key"表示被删除的字段，"value"表示删除的结果；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息

attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

```
{
  "key": "key",
  "value": true
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择删除哈希表中指定字段操作。



状态存储

字符串

列表

哈希

搜索

获取哈希表中指定字段的值

设置哈希表中指定字段的值

删除哈希表中指定字段

检查哈希表中指定字段是否存在

获取哈希表中字段的数量

删除哈希表

设置哈希表的过期时间

2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。删除成功后，message payload 中包含删除的数据：

```
{
  "key": "key",
  "value": true
}
```

检查哈希表中指定字段是否存在

参数配置

参数	数据类型	描述	是否必填	默认值
字段	string	需要检查的字段，长度限制512字符	是	-

通用

字段

string

key

f(x)

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"key"和"value"元素，"key"表示获取的字段，"value"表示检查的结果； 执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

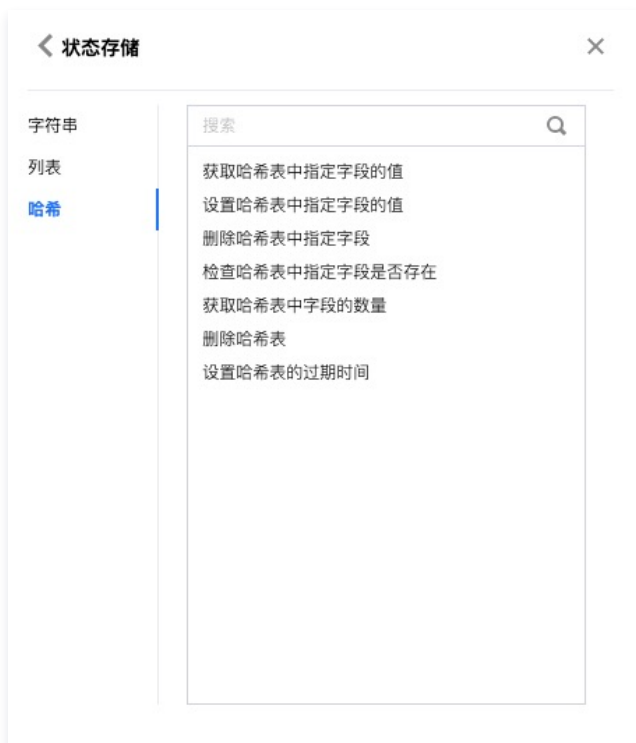
```
{
  "key": "key",
  "value": true
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

- 1. 添加状态存储组件，选择检查哈希表中指定字段是否存在操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。检查成功后，message payload 中包含检查的结果：

```
{
  "key": "key",
  "value": true
}
```

获取哈希表中字段的数量

参数配置

仅需配置通用配置。

输出

执行成功时，输出结果会保存在 message 消息体的 payload；执行失败时，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含"size"元素，"size"表示获取的数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：执行成功后，message payload 值如下：

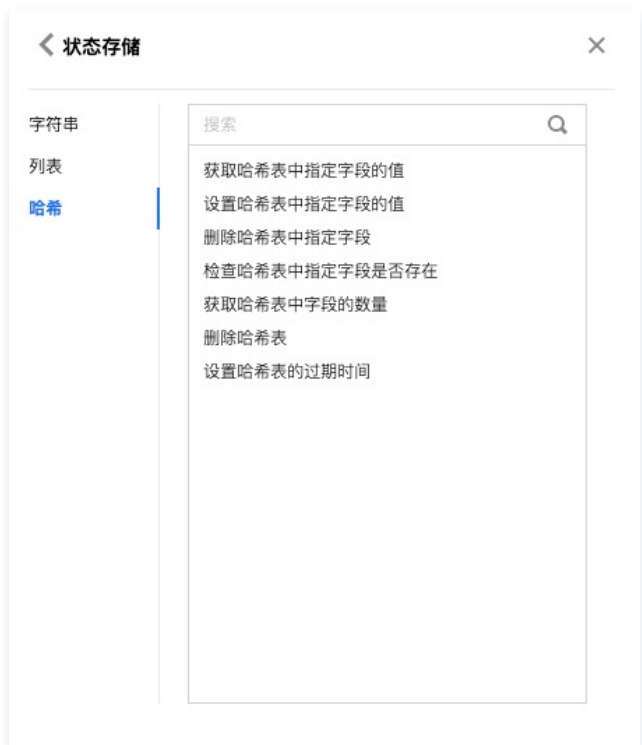
```
{
  "size": 1
}
```

执行失败后，message error 值如下：


```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择获取哈希表中字段的数量操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中包含哈希表中字段的数量：

```
{
  "size": 1
}
```

删除哈希表

参数配置

仅需配置通用配置。

输出

执行成功时，无结果返回；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

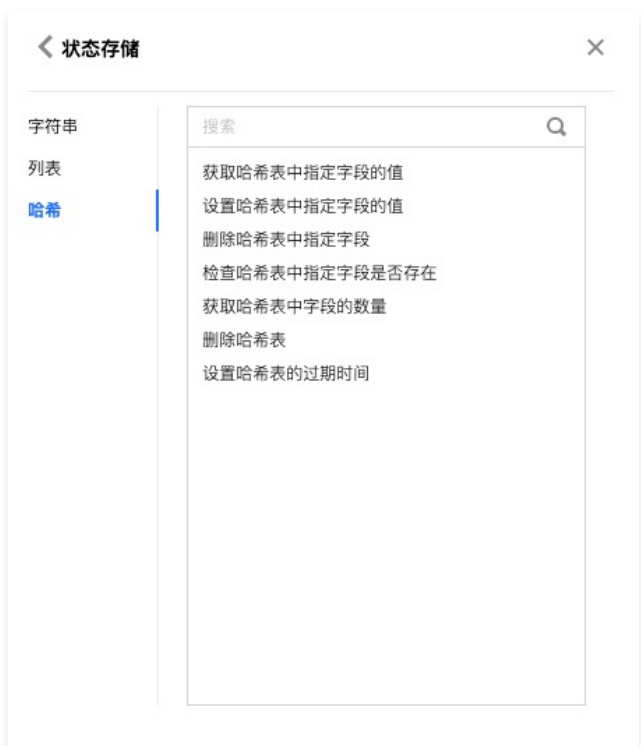
message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Can't connect to MySQL server on '*' "
}
```

案例

1. 添加状态存储组件，选择删除哈希表操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

×

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

×

✓ 选择连接器

>

2 配置连接器

键 *

string

请输入字符串

f(x)

上一步

保存

3. 执行操作。获取成功后，message payload 中为空。

设置哈希表的过期时间

参数配置

参数	数据类型	描述	是否必填	默认值
过期时间	int	过期时间（单位天）	是	-

通用

过期时间(天) *

int

1

f(x)

输出

执行成功时，没有结果返回；执行失败时，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为空；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

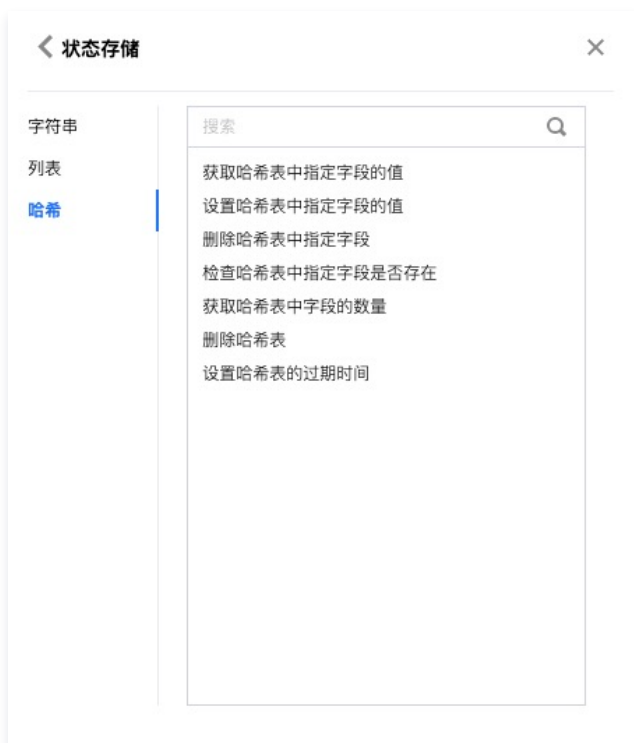
执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
```

```
"Description": "Can't connect to MySQL server on '11'"
}
```

案例

1. 添加状态存储组件，选择设置哈希表的过期时间操作。



2. 新建状态存储配置，配置键。

编辑连接器配置

×

1 选择连接器

>

2 配置连接器

连接器类型 *

状态存储

▼

配置名称 *

请输入配置名称

下一步

取消

编辑连接器配置

×

✓ 选择连接器

>

2 配置连接器

键 *

string

▼

请输入字符串

f(x)

上一步

保存

3. 执行操作。设置成功后，message payload 中为空。

Neo4j 连接器

最近更新时间：2025-04-25 15:16:32

简介

Neo4j 数据库是开源的高性能 NOSQL 图形数据库，图数据库用来解决现有关系数据库的局限性，图模型明确地列出数据节点之间的依赖关系，可以简单快速地检索难以在关系系统中建模的复杂层次结构。Neo4j 数据库适合处理复杂关系，应用在欺诈检测、推荐系统、社交网络图、身份和访问管理等领域。

Neo4j 设计 CQL（Cypher Query Language）查询语言，Neo4j 通过 CQL 可以进行 Neo4j 数据库的增删改查等相关操作，CQL 的语法类似于 SQL，简单易读。

Neo4j 连接器可连接第三方 Neo4j 数据库系统并执行节点的增删改查和 CQL 操作。用户通过连接器配置来配置数据库的连接参数，配置成功后便可执行对应的数据库操作。

连接器配置

配置参数

参数	数据类型	描述	是否必填	默认值
连接URL	string	用于连接 Neo4j 数据库的 URL	是	—
用户名	string	用于连接验证的用户名	否	—
密码	string	用于连接验证的用户密码	否	—
数据库	string	连接的数据库名称	否	—
连接超时时间	int	连接超时时间（单位秒），最长超时时间为30秒	否	5

配置界面

编辑连接器配置

✓ 选择连接器

>

2 配置连接器

通用

连接配置

连接URL *

neo4j://:7687

用户名

neo4j

密码

.....

数据库

string

请输入字符串

f(x)

执行操作的数据库

连接超时时间 (秒)

int

5

最长超时时间：30秒

网关配置

安全网关

无

测试连接

上一步

保存

操作说明

Neo4j 连接器目前支持创建节点、查询节点、删除节点、更新节点、执行 CQL 语句等操作。

创建节点				
参数配置				
参数	数据类型	描述	是否必填	默认值
节点标签名称	string	待创建节点的标签名称	是	-
节点属性	dict	待创建节点的属性，key 为属性名称，value 为属性值	否	-

Neo4j 创建节点

当前使用的是 neo4j-test 物理配置 编辑 | 切换

配置 调试 备注

通用

节点标签名称 *

string 请输入字符串 f(x)

节点属性

☒ 列表添加 ☐ f(x)

value 操作

+添加

节点属性信息，key为属性名称，value为属性值

输出

查询操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 int 类型，返回创建成功的节点数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Neo4j 连接器组件，选择创建节点操作。

Neo4j

默认

搜索

创建节点

查询节点

删除节点

更新节点

执行CQL语句

2. 新建连接器配置，填写配置参数，单击**测试连接**，测试连接器配置是否正确。

编辑连接器配置

✓ 链接配置正确

✓ 选择连接器

>

2 配置连接器

通用

连接配置

连接URL *

neo4jz.2:7687

用户名

neo4j

密码

数据库

string

请输入字符串

f(x)

执行操作的数据库

连接超时时间 (秒)

int

5

最长超时时间: 30秒

网关配置

安全网关

无

测试连接

上一步

保存

3. 在通用配置中，填写节点标签名称和节点属性信息。例如：标签名称填写为 person、节点属性 item_1 的 key 为 name、value 为 'xiaoming'、item_2 的 key 为 age、value 为18。

Neo4j

创建节点

...

当前使用的是 neo4j-test 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

节点标签名称 *

string

person

f(x)

节点属性

☒ 列表添加

☐ f(x)

value	操作
item_1	删除
item_2	删除
+添加	

节点属性信息，key为属性名称，value为属性值

节点属性：

item_1

☒ 手动输入

☐ f(x)

key *

string

name

f(x)

value *

string

xiaoming

f(x)

确定

取消

item_2

☒ 手动输入

☐ f(x)

key *

string

age

f(x)

value *

int

18

f(x)

确定

取消

4. 创建成功后，message payload 返回创建节点的数量。

输出消息

payload

▼

1

下载

5. 若执行过程中出现错误，message error中会包含错误信息。

输出消息

error

▼

```
{
  "Code": "CORE:RUNTIME",
  "Description": "Neo4jError:
Neo.ClientError.Security.Unauthorized (The client is
unauthorized due to authentication failure.)"
}
```

下载

查询节点

参数配置

参数	数据类型	描述	是否必填	默认值
节点标签名称	string	待查询的节点标签名称	是	-
节点属性	dict	节点属性，根据节点属性作为查询过滤条件，不填则返回标签下所有节点	否	-

Neo4j 查询节点

当前使用的是 neo4j-test 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

节点标签名称

string 请输入字符串 f(x)

节点属性

列表添加

f(x)

value

操作

+添加

节点属性信息，key为属性名称，value为属性值

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 list 类型，list 成员为 dict 类型，表示节点的属性信息；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

例如：查询操作执行成功后，查询返回两个节点的属性信息，message payload 值如下：

```
[
  {
    "age": 18,
    "name": "xiaohua"
  },
  {
    "age": 18,
    "name": "xiaoming"
  }
]
```

案例

1. 添加 Neo4j 连接器组件，选择查询节点操作。



2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入节点标签名称和节点属性。例如：标签名称填写为 person、节点属性 item_1 的 key 值为 age、value 值为18，即可筛选出 age 属性值为18的节点。

item_1

☒ 手动输入 ☐ f(x)

key *

string ▼

age

f(x)

value *

int ▼

18

f(x)

确定

取消

4. 查询成功后，message payload 中包含执行结果，返回 age 属性为18的两个节点信息。



更新节点

参数配置

参数	数据类型	描述	是否必填	默认值
节点标签名称	string	待删除的节点标签名称	是	-
节点属性	dict	节点属性，根据节点属性作为删除条件，不填则更新标签下所有节点	否	-
更新后的节点属性	dict	更新后的节点属性	是	-

Neo4j 更新节点

当前使用的是 neo4j-test 物理配置

配置 调试 备注

通用

节点标签名称 *

string 请输入字符串 f(x)

节点属性 *

列表添加 f(x)

value 操作

+添加

节点属性信息，key为属性名称，value为属性值

更新后的节点属性

列表添加 f(x)

value 操作

+添加

更新后的节点属性信息，key为属性名称，value为属性值

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 Bool 类型，更新成功为 true，更新失败为 false；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Neo4j 连接器组件，选择更新节点操作。



2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入所需参数。例如：节点标签填写为 person、节点属性 item_1 的 key 值为 age、value 值为18，更新后的节点属性采用 Dataway 表达式输入。

```
def dw_process(msg):  
    return {  
        'name': 'abc',  
        'age': 30  
    }  
}
```

4. 执行成功后，message payload 中包含执行结果，true 表示更新成功。



- 再执行查询节点操作，可以看到两个节点已经完成更新。



删除节点

参数配置

参数	数据类型	描述	是否必填	默认值
节点标签名称	string	待删除的节点标签名称	是	-
是否删除节点关系	bool	是否删除节点间的关系信息	否	true
节点属性	dict	节点属性，根据节点属性作为删除条件，不填则删除标签下所有节点	否	-

Neo4j 删除节点

当前使用的是 neo4j-test 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

节点标签名称 *

string

请输入字符串

f(x)

是否删除节点关系

bool

true

f(x)

节点属性

string

请输入字符串

f(x)

节点属性信息，key为属性名称，value为属性值

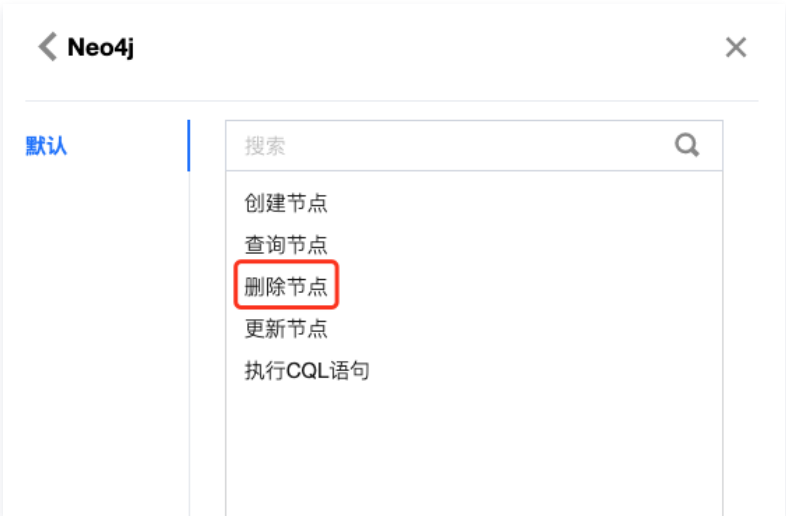
输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 int 类型，代表删除的节点数量；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 元素：“Code” 表示错误类型，“Description” 表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Neo4j 连接器组件，选择删除节点操作。



2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填写相关参数。例如：节点标签填写为 person、节点属性采用 Dataway 表达式输入，输入如下：

```
def dw_process(msg) :  
    return {  
        'age': 30  
    }  
}
```

4. 删除成功后，message payload 中包含成功删除的节点数量。



执行CQL语句

参数配置

参数	数据类型	描述	是否必填	默认值
CQL语句	string	待执行的 CQL 语句，遵循 Cypher Query Language 标准的语句，支持原生 CQL 及参数化输入。参数化使用(\$+参数名称)输入	-	-
输入参数	string	CQL 语句的输入参数，key 为参数名称，参数名称需要和 input 参数名称一致，value 为参数值	是	-

Neo4j 执行CQL语句

当前使用的是 neo4j-test 物理配置

编辑 | 切换

配置 | 调试 | 备注

通用

CQL语句

string MATCH (n:Person {name:\$name}) | f(x)

遵循Cypher Query Language标准的语句，支持原生CQL及参数化输入。参数化使用(\$+参数名)输入

输入参数

列表添加

f(x)

value

操作

+添加

请填入CQL语句中的参数，key为参数名称，参数名称需要和input参数名称一致，value为参数值

输出

操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。

组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 为 dict 类型，包含4个字段：selectResult、createCount、deleteCount、containsUpdates。 - selectResult 字段代表查询结果，对应的值为 list 类型，list 成员为 dict 类型，为节点的属性信息 - createCount 代表创建的节点数量，对应的值为 int 类型 - deleteCount 代表删除的节点数量，对应的值为 int 类型 - containsUpdates 代表是否执行更新，对应的值为 bool 类型；执行失败后，payload 为空
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含“Code”和“Description”元素：“Code”表示错误类型，“Description”表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

案例

1. 添加 Neo4j 连接器组件，选择执行 CQL 操作。



2. 新建连接器或选择已创建的连接器。
3. 在通用配置中，填入相关信息。例如：“CQL 语句”为 `MATCH (n:person {age:$age}) RETURN n`，CQL 语句中包含 `age` 参数变量，输入参数采用 Databay 表达式输入，如下：

```
def dw_process(msg):  
    return {  
        'age': 18  
    }  
}
```

○ 界面如图：



4. 执行成功后，message payload 中包含执行结果，由于执行的是查询命令，“selectResult” 字段包含查询结果。

输出消息 payload ▾

```
{
  "containsUpdates": false,
  "createCount": 0,
  "deleteCount": 0,
  "selectResult": [
    {
      "age": 18,
      "name": "xiaohua"
    },
    {
      "age": 18,
      "name": "xiaoming"
    }
  ]
}
```

JWT 连接器

最近更新时间：2023-06-08 14:48:03

简介

JWT（Json Web Token）是一种用于在网络应用环境间传递声明而执行的一种基于 JSON 的开放标准（[RFC 7519](#)）。JWT 被设计为紧凑且安全的，适用于分布式站点的单点登录（SSO）场景。Json Web Token 的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可用于增加一些额外的其它业务逻辑所必须的声明信息，Json Web Token 也可直接被用于认证，也可被加密。
JWT 连接器提供生成 Json Web Token 以及校验 Json Web Token 的相关功能。

配置

配置参数：

参数	数据类型	描述	是否必填	默认值
签名方法	enum	NONE/HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512/RSA-SHA-256/RSA-SHA-384/RSA-SHA-512/RSAPSS-SHA-256/RSAPSS-SHA-384/RSAPSS-SHA-512	是	无

选择“NONE”时，无配置参数，如下图：

新建连接器配置

选择连接器

配置连接器

签名方法

NONE

f(x)

上一步

保存

选择“HMAC-SHA-256/HMAC-SHA-384/HMAC-SHA-512”时，配置参数如下：

参数	数据类型	描述	是否必填	默认值
HMAC 密钥	string	HMAC密钥	是	无

新建连接器配置

选择连接器

配置连接器

签名方法

HMAC-SHA-256

f(x)

HMAC密钥

上一步

保存

选择”RSA-SHA-256/RSA-SHA-384/RSA-SHA-512/RSAPSS-SHA-256/RSAPSS-SHA-384/RSAPSS-SHA-512 “时，配置参数如下：

参数	数据类型	描述	是否必填	默认值
私钥证书文件	string	RSA 私钥证书文件	是	无
公钥证书文件	string	RSA 公钥证书文件	是	无

新建连接器配置

×

✓ 选择连接器

>

2 配置连接器

签名方法

RSA-SHA-256

▼

f(x)

私钥证书文件

还未选择文件

选择文件

公钥证书文件

还未选择文件

选择文件

上一步

保存

操作说明

JWT 组件目前支持生成 JWT 和校验 JWT 操作。

生成 Json Web Token

参数配置

参数	数据类型	描述	是否必填	默认值
签发者（iss）	string	JWT 签发者	否	无
主题（sub）	string	JWT 主题	否	无
唯一身份标识 ID（jti）	string	JWT 的唯一身份标识 ID	否	无
接收者列表（aud）	[]string	接收 JWT 的对象列表	否	无
签发时间（iat）	datetime	JWT 的签发时间	否	无
有效起始时间（nbf）	datetime	JWT 的有效起始时间，定义在该时间之前，JWT 都是不可用的	否	无
有效时间（exp）	int	有效起始时间加上有效时间为 JWT 的过期时间	否	3600（单位：秒）
自定义声明	dict	JWT 自定义声明	否	无

JWT 生成JsonW...

...

当前暂未绑定任何配置，请选择或者新建连接配置

新建 | 绑定

配置 | 调试 | 备注

标准声明配置

签发者(iss)

string

请输入字符串

f(x)

主题(sub)

string

请输入字符串

f(x)

唯一身份标识ID(jti)

string

请输入字符串

f(x)

接收者列表(aud)

☐ 表达式

value

操作

string

请输入字符串

f(x)

添加

删除

签发时间(iat)

datetim

2000-01-01 00:00:00

f(x)

有效起始时间(nbf)

datetim

2000-01-01 00:00:00

f(x)

有效时间(秒)(exp)

int

3600

f(x)

自定义声明配置

自定义声明

☐ 表达式

value

操作

记录1

☐ 表达式

删除

参数名称

string

请输入字符串

f(x)

参数值

string

请输入字符串

f(x)

输出

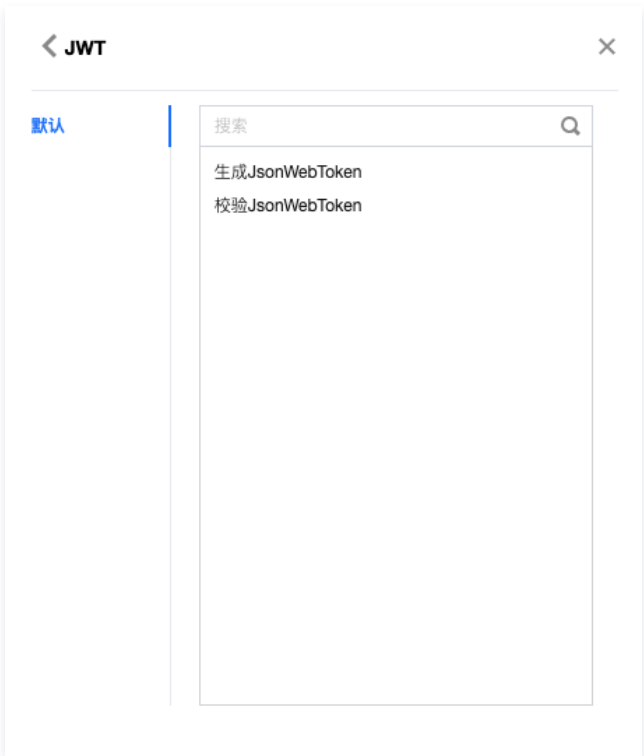
操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
payload	- 执行成功后，payload 为 binary 类型，返回生成的 JWT 内容。 - 执行失败后，payload 为空。
error	- 执行成功后，error 为空。 - 执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。

variable	继承上个组件的 variable 信息。
----------	----------------------

案例1

1. 添加 JWT 组件，选择生成 JsonWebToken 操作：



2. 在连接器配置中，填写相关参数：

编辑连接器配置

签名方法

HMAC-SHA-512

f(x)

HMAC密钥

保存

3. 在操作配置填写参数如下：

JWT 生成JsonW...

...

当前使用的是 jwt2 连接配置 编辑 | 切换

配置

备注

标准声明配置

签发者(iss)

string

user

f(x)

主题(sub)

string

client

f(x)

JWT ID(jti)

string

id222

f(x)

接收者(aud)

☐ 表达式

value

操作

string

sombody_else

f(x)

添加

删除

签发时间(iat)

datetim

2000-01-01 00:00:00

f(x)

生效时间(nbf)

datetim

2000-01-01 00:00:00

f(x)

有效时间(exp)

int

3600

f(x)

(单位:秒)

自定义声明配置

自定义声明

☐ 表达式

value

操作

记录1

☐ 表达式

删除

参数名称

string

name

f(x)

参数值

string

wsu

f(x)

+添加

腾讯云

腾讯轻联

版权所有：腾讯云计算（北京）有限责任公司

第636 共732页

输出消息

payload

复制

"eyJhbGciOiJIUzUxMiIsInR5cCI6IkpXVCJ9.eyJhdWQiOiBzic29tZWJvZHIzWwzZSjdLCJleHAiOiE2Mzg1MTk1MzIsImhldCI6MTYzODUxNTkzMiwiaXNzIjoidXNlcisImP0aSI6ImlkMjlyIiwibmFzI6IjndzdylsIm5iZiI6MTYzODUxNTkzIiwic2ViljoiY2xpZW50In0.8l-lv5osKhlJlCwB6BaK6HmFjexQjGlyKDigCdQmmPxRCTnyk6Xp8tLf7oaUlGSQl_Ui6DKjWcqbQA8eA"

下载

参数配置

参数	数据类型	描述	是否必填	默认值
Token	string	生成的 Json Web Token	是	无
接收者列表（aud）	[]string	用于校验接收 JWT 的对象是否匹配	否	无

 JWT  校验JsonW...

...

□

当前暂未绑定任何配置，请选择或者新建连接配置

配置

调试

备注

通用配置

Token *

string ▼

请输入字符串

f(x)

接收者列表
(aud)

☐ 表达式

value

操作

string ▼

请输入字符串

f(x)

添加

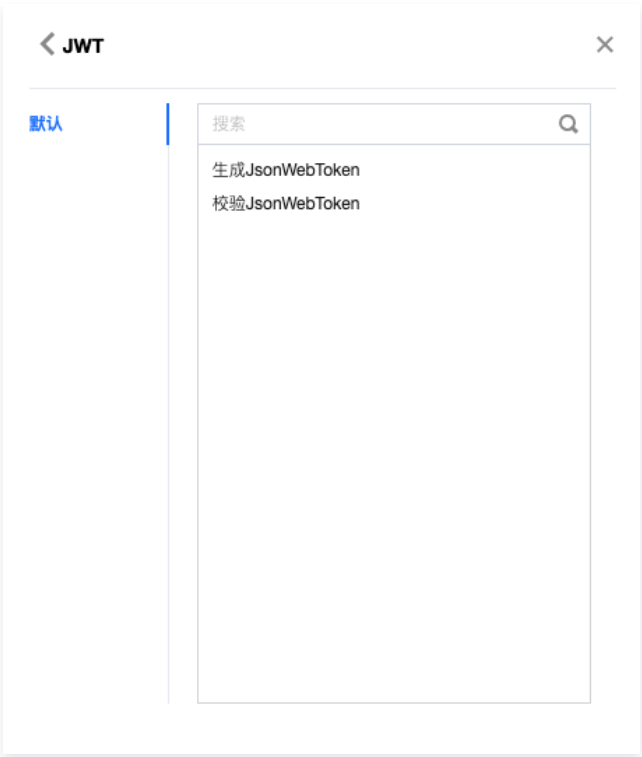
删除

操作执行成功后，输出结果会保存在Message消息体的payload；执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message属性	值
payload	• 执行成功后，payload 为 binary 类型，返回校验解析 JWT 后的内容。 • 执行失败后，payload 为空。
error	• 执行成功后，error 为空。 • 执行失败后，error 为 dict 类型，包含“Code”和“Description”字段：“Code”字段表示错误类型，“Description”字段表示错误具体信息。
attribute	继承上个组件的 attribute 信息。
variable	继承上个组件的 variable 信息。

案例2

1. 添加 JWT 组件，选择校验 JsonWebToken 操作



2. 在连接器配置中，填写相关参数：



3. 在操作配置填写参数如下：

JWT ? 校验JsonW...

...

当前使用的是 jwt2 连接配置

编辑 | 切换

配置

备注

通用配置

Token *

string

def dw_process(msg): return ms{

接收者(aud)

☐ 表达式

value

操作

string

sombody_else

f(x)

添加 删除

高级配置

输出消息绑定自定义数据类型

消息属性

payload

类型选择

空

+

4. 执行成功后，message payload 中为校验解析后的 JsonWebToken 内容：

输出消息 payload

复制

```

{
  "aud": [
    "sombody_else"
  ],
  "exp": "2021-12-03 16:18:52",
  "iat": "2021-12-03 15:18:52",
  "iss": "user",
  "jti": "id222",
  "name": "wsw",
  "nbf": "2021-12-03 15:18:52",
  "sub": "client"
}

```

下载

GRPC 连接器

最近更新时间：2023-06-08 14:48:03

简介

GRPC:Request 连接器作为 GRPC 客户端，可以发起 GRPC 请求并将得到的响应作为消息传递给下一个组件。

操作说明

输入参数

GRPC:Request 操作配置包括基本配置、服务定义、GRPC 参数和安全网关配置：

GRPC 配置

配置 调试 备注

基本配置

请求域名

string

请输入字符串

f(x)

请求端口

int

请输入整数

f(x)

认证机制

No Auth

启用SSL/TLS

bool

false

f(x)

请求体

请输入表达式

服务定义

Proto文件

还未选择文件

选择文件

f(x)

Proto文件过期时间(秒)

int

300

f(x)

服务名称

string

请输入字符串

f(x)

方法名称

string

请输入字符串

f(x)

GRPC参数

保活心跳间隔时间(秒)

int

60

f(x)

保活心跳确认超时时间(秒)

int

20

f(x)

是否允许发送保活心跳

bool

false

f(x)

连接超时时间(秒)

int

60

f(x)

最大消息大小(MB)

int

5

f(x)

负载均衡策略

Round Robin

f(x)

安全网关配置

安全网关

无

说明

您可以单击以下页签，查看操作配置中涉及到的不同模块的参数描述。

基本配置

参数	数据类型	描述	是否必填	默认值
请求域名	string	对应的请求域名	是	无
请求端口	int	请求端口	是	无
认证机制	enum	默认为 No Auth，目前支持 Basic Auth，如果为 Basic Auth，则需配置用户名、密码	否	No Auth
用户名	string	用户名	否	无
密码	string	密码	否	无
启用 SSL/TLS	bool	是否启用 SSL/TLS	否	false
服务器证书文件	file	服务器证书文件	否	无
请求体	entity	GRPC 请求发送的请求体	是	无

服务定义

参数	数据类型	描述	是否必填	默认值
Proto 文件	file	GRPC 请求的服务端的 proto 文件	是	无
Proto 文件过期时间（秒）	int	Proto 文件的缓存过期时间	否	300
服务名称	string	GRPC 请求的服务名称	是	无
方法名称	string	GRPC 请求的方法名称	是	无

GRPC 参数

参数	参数类型	描述	是否必填	默认值
保活心跳间隔时间（秒）	int	GRPC 请求的保活心跳间隔时间，最小为10	否	60
保活心跳确认超时时间（秒）	int	GRPC 请求的保活心跳确认超时时间	否	20
是否允许发送保活心跳	bool	是否允许发送保活心跳	否	false
连接超时时间（秒）	int	连接超时时间	否	60
最大消息大小（MB）	int	GRPC 请求发送和接收的最大消息大小	否	5
负载均衡策略	enum	GRPC 请求的负载均衡策略	否	Round Robin

安全网关配置

参数	数据类型	描述	是否必填	默认值
安全网关	string	直接请求内网域名/IP 时，需要绑定安全网关	否	无

输出参数

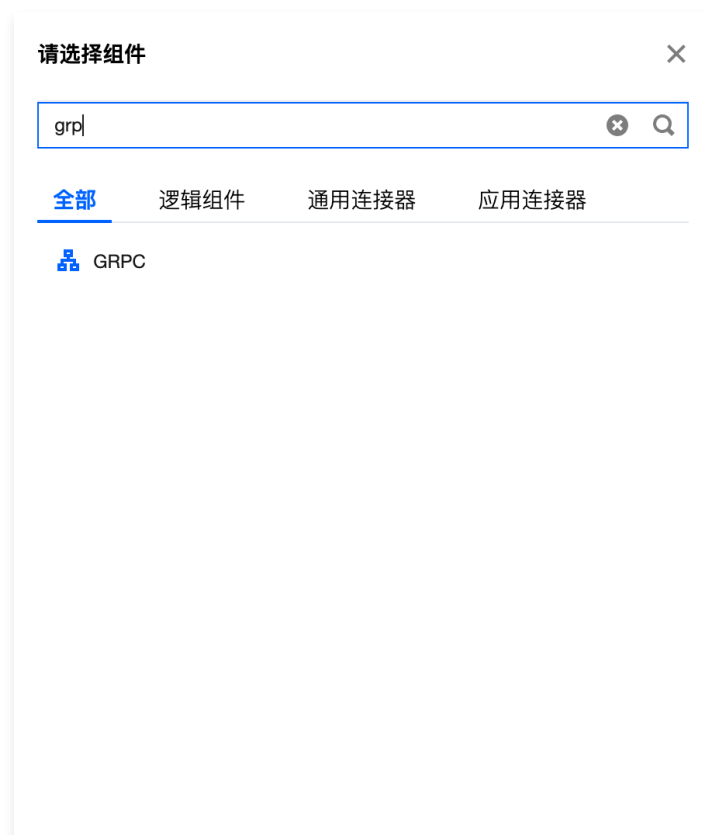
GRPC:Request 请求的响应返回后，会将响应结果生成对应的消息传递给下一个组件。其中，响应的 body 会放到消息的 payload 中。

GRPC:Request 输出消息描述：

消息属性	值
payload	响应 body 保存在 payload 中
error	执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attributes	默认为空
variables	默认为空

案例

1. 组件筛选 GRPC:Request。



2. 新建 GRPC:Request 连接器。



3. 使用操作配置：

GRPC ? GRPC Req...

配置

调试

备注

基本配置

请求域名 *

string

dd.com.cn.resource.ipaaS.mycbk

f(x)

请求端口 *

int

1949

f(x)

认证机制

Basic Auth

用户名

123

密码

...

启用SSL/TLS

bool

true

f(x)

服务器证书文件

d34dd138-b463-45a0-83ea-eba

选择文件

f(x)

请求体 *

def dw_process(msg): return {"name": "bot"

服务定义

Proto文件 *

b795cf99-fbad-4214-9d27-0a0d

选择文件

f(x)

Proto文件过期时间(秒)

int

10

f(x)

服务名称 *

string

HelloServer

f(x)

方法名称 *

string

SayHello

f(x)

GRPC参数

保活心跳间隔时间(秒)

int

10

f(x)

保活心跳确认超时时间(秒)

int

10

f(x)

是否允许发送保活心跳

bool

false

f(x)

连接超时时间(秒)

int

10

f(x)

最大消息大小(MB)

int

5

f(x)

负载均衡策略

Round Robin

f(x)

安全网关配置

安全网关

无

4. 发布并触发：

GRPC ? GRPC Req...

配置

调试

备注

运行消息

历史消息

2021-11-19
19:22:34.580

2021-11-19
18:59:53.665

2021-11-19
17:42:41.111

2021-11-19
16:24:18.0

2021-11-19
16:23:30.932

2021-11-19
14:23:42.335

2021-11-19
14:22:9.624

2021-11-19
12:38:38.985

2021-11-19
12:28:12.676

2021-11-19
12:27:28.519

输入消息 payload 复制

""

下载

输出消息 payload 复制

```
{  
  "msg": "hello"  
}
```

下载

MQTT 连接器

最近更新时间：2023-06-08 14:48:03

简介

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议），是一种基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议，该协议构建于 TCP/IP 协议上。MQTT 最大优点在于，可以以极少的代码和有限的带宽，为连接远程设备提供实时可靠的消息服务。作为一种低开销、低带宽占用的即时通讯协议，使其在物联网、小型设备、移动应用等方面有较广泛的应用。

MQTT 目前主流使用的协议版本是 MQTT V3.1.1版本，iPaaS MQTT 连接器基于该协议实现，可使用 MQTT Consumer 操作从 MQTT Broker 中消费消息，或通过 MQTT Producer 操作，将各类 SaaS 应用产生的消息发布到指定的 Broker 上。

连接器配置

通用配置标签页

参数	数据类型	描述	是否必填	默认值
Broker URL	string	MQTT Server Broker 地址，格式为 ip:port/path或url:port/path	是	无
连接协议	enum	选择连接协议类型，需和 port 对应	是	TCP
Client ID	enum	客户端 ID	是	无
用户名	string	MQTT 用户名	否	无
密码	string	MQTT 密码	否	无
TLS 客户端证书	file	使用提供的客户端证书对连接进行加密，仅当连接协议为 TLS 或 WebSocket Secure 时可配置	否	无
TLS 客户端 Key	file	客户端证书文件对应的 Key 文件，仅当连接协议为 TLS 或 WebSocket Secure 时可配置	否	无
TLS 服务端证书	file	使用提供的服务端证书对连接进行加密，仅当连接协议为 TLS 或 WebSocket Secure 时可配置	否	无

编辑连接器配置

通用配置

Broker URL *

string

def dw_process(msg): return "br"

MQTT Server Broker地址，格式为ip:port/path或url:port/path

连接协议

TCP

f(x)

连接协议类型和Broker URL的端口号应匹配

Client ID *

string

tucdvfdvf0sdvsf0bjjfv

f(x)

用户名

string

请输入字符串

f(x)

密码

f(x)

高级配置标签页

参数	数据类型	描述	是否必填	默认值
是否启用遗嘱消息	bool	无	否	false
遗嘱主题	string	启用遗嘱后，可填写	否	无
遗嘱内容	string	启用遗嘱后，可填写	否	无
遗嘱 QoS	enum	启用遗嘱后，可填写	否	无
是否保存遗嘱消息	bool	启用遗嘱后，可填写	否	无
是否清除会话	bool	clean session	否	true
是否顺序发送	bool	order	否	false
keep-alive 超时时间	string	无	否	无
ping 超时时间	string	无	否	无
connect 超时时间	string	无	否	无
是否自动重试连接	bool	auto-reconnect	否	false
reconnect 最大间隔时间	string	仅当自动重试连接为true时可填写	否	无
是否重试 connect	bool	retry connect	否	false
connect 重试间隔时间	string	仅当重试connect为true时可填写	否	无
write 超时时间	string	无	否	无

编辑连接器配置

✕

是否启用遗嘱消息

bool

false

▼

f(x)

是否清除会话

bool

true

▼

f(x)

clean session

是否顺序发送

bool

false

▼

f(x)

keep-alive超时时间(毫秒)

string

请输入字符串

▼

f(x)

ping超时时间(毫秒)

string

请输入字符串

▼

f(x)

connect超时时间(毫秒)

string

请输入字符串

▼

f(x)

是否自动重试连接

bool

false

▼

f(x)

是否重试 connect

bool

false

▼

f(x)

write超时时间(毫秒)

string

请输入字符串

▼

f(x)

测试连接

保存

操作配置

MQTT 连接器包含 Consumer（消费消息）和 Producer（生产消息）两种操作。

Consumer 操作

输入参数

参数	数据类型	说明	是否必填	默认值
订阅主题	Array Of String	订阅的队列名称列表	是	无
QoS	enum	At Most Once	否	无
确认模式	enum	消息消费后的应答模式，有两种： - 消费后直到触发的流成功结束后才确认 offset - 消费后立即确认offset，不等待流的触发及运行结果	流运行成功后确认	true

MQTT MQTT Con...

当前使用的是 mqtt-v3 连接配置

编辑 | 切换

配置备注

通用

通用配置

订阅主题

☐ 表达式

value

操作

string

ipaaS-test-topic

f(x)

添加

删除

高级配置

QoS

At most once

f(x)

确认模式

流运行成功后确认

f(x)

输出参数

Consumer 操作执行成功后，输出结果会保存在 message 消息体的 payload；执行失败后，错误信息会保存在 message 消息体的 error。组件输出的 message 信息如下：

message 属性	值
payload	执行成功后，payload 的 raw data 为 byte 数组
error	- 执行成功后，error 为空 - 执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段：“Code” 字段表示错误类型，“Description” 字段表示错误具体信息
attribute	执行成功后，attribute 为 dict， 查看属性列表
variable	继承上个组件的 variable 信息

atrributes 属性列表：

- QoS: int, 0: At most once, 1: At least once, 2: Exactly once
- topic: string, 消费到的消息所属的 Topic
- messageID: uint16, 消息 ID

例如，执行成功后，message payload 的 raw data 值为 byte 数组，message attribute 值如下：

```
{
  "QoS": 1,
  "topic": "some topic",
  "messageID": 4
}
```

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "some error message."
}
```

Producer 操作

输入参数（通用）

参数	数据类型	说明	是否必填	默认值
主题	string	消息内容将发布到指定的主题	是	无
消息内容	string	无	否	无
QoS	enum	无	否	At Most Once
是否保存消息	bool	无	否	false

MQTT MQTT Prod...

当前使用的是 mqtt-v3 连接配置 编辑 | 切换

配置 备注

通用

通用配置

主题

string

ipaaS-mqtt-test-topic

f(x)

消息内容将发布到指定的主题

消息内容

string

hello from ipaaS

f(x)

高级配置

QoS

At most once

f(x)

是否保存消息

bool

false

f(x)

输出参数

Publish 操作执行失败后，错误信息会保存在 message 消息体的 error。
组件输出的 message 信息如下：

message 属性	值
------------	---

payload	继承上个组件的 payload 信息
error	- 执行成功后，error为空 - 执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	继承上个组件的 attribute 信息
variable	继承上个组件的 variable 信息

执行失败后，message error 值如下：

```
{
  "Code": "CORE:RUNTIME",
  "Description": "some error message."
}
```

MySQL Change Data Capture 触发器

最近更新时间：2023-06-08 14:48:03

简介

MySQL Change Data Capture 连接器可连接第三方MySQL数据库系统并监控binlog中的 **DML** 操作，捕获 MySQL 数据库 DML 操作记录并触发 iPaaS 集成流，完成相应的集成业务逻辑。

类型映射关系

MySQL 数据类型和 MySQL Change Data Capture 连接器捕获到的 DML 记录中的数据类型映射关系如下表。

MySQL 类型	MySQL Change Data Capture 连接器类型
BOOLEAN, BOOL	BOOLEAN
BIT (1)	BOOLEAN
BIT (>1)	BYTES
TINYINT	INT16
SMALLINT [(M)]	INT16
MEDIUMINT [(M)]	INT32
INT, INTEGER [(M)]	INT32
BIGINT [(M)]	INT64
REAL [(M,D)]	FLOAT32
FLOAT [(M,D)]	FLOAT64
DOUBLE [(M,D)]	FLOAT64
CHAR (M)]	STRING
VARCHAR (M)]	STRING
BINARY (M)]	BYTES or STRING
VARBINARY (M)]	BYTES or STRING
TINYBLOB	BYTES or STRING
TINYTEXT	STRING
BLOB	BYTES or STRING
TEXT	STRING
MEDIUMBLOB	BYTES or STRING
MEDIUMTEXT	STRING
LOB	BYTES or STRING
LONGTEXT	STRING
JSON	STRING
ENUM	STRING
SET	STRING

<code>`YEAR[(2\</code>	<code>4)]`</code>
<code>TIMESTAMP [(M)]</code>	<code>STRING</code>

MySQL 配置

需要在 MySQL Server 端进行必要的配置并且生效后，才可以使用 MySQL Change Data Capture 连接器，否则 MySQL Change Data Capture 连接器将无法捕获到 DML 操作并触发流。

以 Linux 环境为例说明如何配置 MySQL。

1. 开启 Binlog。

1.1 使用命令行工具连接到 MySQL 数据库所在服务器，执行一下命令以 root 用户登录数据库。

```
mysql -uroot -ppassword
```

1.2 执行以下命令，查询 MySQL 数据库是否开启了 Binlog。

```
show variables like 'log_bin';
```

- 若变量 `log_bin` 的值为“OFF”，则说明 Binlog 未开启，继续执行下一步。
- 若变量 `log_bin` 的值为“ON”，则说明 Binlog 已开启，继续执行以下 SQL 命令，检查相关参数的配置是否符合要求。

```
show variables like '%binlog_format%';  
show variables like '%binlog_row_image%';
```

- 变量 `binlog_format` 的值应该为“ROW”，变量 `binlog_row_image` 的值应该为“FULL”。如果满足要求，直接跳到 2，否则继续执行下一步。

1.3 执行以下命令退出数据库。

```
exit;
```

1.4 执行以下命令编辑 MySQL 配置文件，然后按 `i` 进入输入模式。

```
vi /etc/my.cnf
```

1.5 在配置文件中增加如下配置，开启 Binlog。

```
server-id = 123  
log_bin = mysql-bin  
binlog_format = row  
binlog_row_image = full  
expire_logs_days = 14  
gtid_mode = on  
enforce_gtid_consistency = on
```

其中：

- **server-id** 的值应为大于1的整数，请根据实际情况设置。
- **expire_logs_days** 为 Binlog 日志文件保留时间，超过保留时间的 Binlog 日志会被自动删除，应保留至少2天的日志文件。
- **gtid_mode = on** 和 **enforce_gtid_consistency = on** 仅当 MySQL 的版本大于等于5.6.5时才需要添加，否则删除这两行内容。

1.6 按 `Esc` 退出输入模式，然后输入 `:wq` 并回车，保存退出。执行以下命令重启 MySQL 数据库。

```
service mysqld restart
```

1.7 以 root 用户登录数据库，执行以下命令，查询变量 `log_bin` 的值是否为“ON”，即是否已开启 Binlog。

```
show variables like 'log_bin';
```

2. 在数据库中执行以下命令创建数据连接器连接数据库的用户并配置权限。

```
CREATE USER 'ipaas'@'%' IDENTIFIED BY 'password';
GRANT SELECT, RELOAD, SHOW DATABASES, REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'ipaas'@'%';
```

- 其中：
- ipaas 为数据连接器连接用户名，请根据实际情况设置。
 - password 为数据连接器连接用户密码，请根据实际情况设置。

3. 执行以下命令退出数据库连接。

```
exit;
```

连接器配置

MySQL Change Data Capture 连接器配置参数

参数	数据类型	描述	是否必填	默认值
数据库类型	enum	数据库类型，仅支持 MySQL	是	MySQL
数据库 Host 地址	string	MySQL 数据库地址	是	-
端口号	int	MySQL 数据库端口号	是	-
用户名	string	用于连接数据库的用户名	是	-
密码	string	用于连接数据库的用户密码	是	-
安全网关	-	选择用于连接用户内网MySQL的代理，安全网关的配置可参见 安全网关	否	-
消息确认模式	enum	消息确认模式： - 流开始前立即确认：change 消息只会处理一次，无论流运行成功与否 - 流运行成功后确认：change 消息仅在流运行成功才会确认，流运行失败的 change 信息会重试	否	流运行成功后确认

连接器配置界面如下：

新建连接配置

数据库类型

MySQL

数据库Host地址

请输入字符串

端口号

int

请输入整数

端口号范围：0-65535

用户名

请输入字符串

账户名称

密码

账户密码

高级配置

安全网关

无

消息确认模式

流运行成功后确认

测试连接

完成

操作说明

MySQL Change Data Capture 连接器目前仅支持作为 Trigger 节点使用，可将捕获到的 dml 操作作为触发消息源，触发后续的业务逻辑流执行。

参数配置

参数	数据类型	描述	是否必填	默认值
数据库	string	选中的待同步数据库名称，根据连接参数自动从 MySQL 数据库中查询并列出	是	—
表	string	选中待同步的表名称，根据连接参数和选中的数据库名称，自动从 MySQL 数据库中查询并列出	是	—
字段	list	选中的待同步字段名称列表，根据连接参数和选中的数据库名称、表名称，自动从 MySQL 数据库中查询并列出，可多选	是	—

参数配置界面如下：

配置

备注

通用

数据库

cdctest2

表

cdctest1

字段

☒ 字段名

操作

☒ id

删除

☒ k

删除

☒ c

删除

☒ pad

删除

输出

message 属性	值
payload	执行成功后，payload 为 json 序列化后的 byte 数组，struct 为: [{"xxx": "xyz"}, {"xxx": "xyz"}]，列表中每一个记录都是一个 dml 操作的信息
error	执行成功后，error 为空；执行失败后，error 为 dict 类型，包含 “Code” 和 “Description” 字段： “Code” 字段表示错误类型， “Description” 字段表示错误具体信息
attribute	执行成功后，attribute 为 dict，包含一个 key 为 totalCount， value 为 payload 中列表记录总数的键值对
variable	继承上个组件的 variable 信息

payload 示例：

```
[
  {
    "after":{
      "c":"111222dgfgh",
      "id":8,
      "k":8,
      "pad":"abcsghg"
    },
    "before":{
      "c":"111222",
      "id":8,
      "k":8,
      "pad":"abcsghg"
    },
    "op":"u",
    "source":{
      "db":"cdctest2",
      "snapshot":"false",
      "table":"cdctest1"
    },
    "transaction":null,
    "ts_ms":1662712532622
  },
  {
```

```
      "after":{
        "c":"1111",
        "id":9,
        "k":9,
        "pad":"zzz"
      },
      "before":{
        "c":"3333dfsdgh",
        "id":9,
        "k":9,
        "pad":"zzz"
      },
      "op":"u",
      "source":{
        "db":"cdctest2",
        "snapshot":"false",
        "table":"cdctest1"
      },
      "transaction":null,
      "ts_ms":1662712532623
    }
  ]
}
```

payload 说明

Snapshot

发布一条由 MySQL Change Data Capture 连接器触发的流之后，连接器会从保存的 binlog 初始 Offset 位置开始捕获 DML 信息，保存的 Offset 称之为一个 snapshot（快照）。

- 如果 Offset 不存在，连接器会从 binlog 的最初位置开始同步，此时会创建 snapshot，并全量同步数据库状态。
- 如果 Offset 存在，连接器会从该 snapshot 记录的位置开始同步。

采集到的 DML 信息中，按照事件类型可以分为4种：

- Read Event：仅出现在 snapshot 阶段，此时 snapshot 被置为 true。

```
{
  "after":{
    "account_type":"a",
    "acct_id":9999,
    "acct_name":"093592",
    "id":9999
  },
  "before":null,
  "op":"r",
  "source":{
    "connector":"mysql",
    "db":"ipaas_test",
    "file":"mysql-bin.000005",
    "gtid":null,
    "name":"_7ef6d5bd9f7fac34ca74b91088375d03f4b94d3af0aeef61dec9e9743336d19e",
    "pos":154,
    "query":null,
    "row":0,
    "sequence":null,
    "server_id":223344,
    "snapshot":true,
    "table":"account_info",
    "thread":null,
    "ts_ms":1662644525154,
    "version":"1.9.3.Final"
  },
  "transaction":null,
}
```

```
"ts_ms":1662644525154
}
```

- **Create Event: Insert** 操作对应的记录。

```
{
  "op":"c",
  "ts_ms":1465491411815,
  "before":null,
  "after":{
    "account_type":"a",
    "acct_id":9999,
    "acct_name":"093592",
    "id":9999
  },
  "source":{
    "version":"1.9.3.Final",
    "connector":"mysql",
    "name":"_7ef6d5bd9f7fac34ca74b91088375d03f4b94d3af0aeef61dec9e9743336d19e",
    "ts_ms":0,
    "snapshot":false,
    "db":"ipaas_test",
    "table":"account_info",
    "server_id":223344,
    "gtid":null,
    "file":"mysql-bin.000005",
    "pos":154,
    "row":0,
    "thread":7,
    "query":null
  },
  "transaction":null
}
```

- **Update Event: Update** 操作对应的记录。

```
{
  "before":{
    "account_type":"a",
    "acct_id":9999,
    "acct_name":"093592",
    "id":9999
  },
  "after":{
    "account_type":"a",
    "acct_id":9999,
    "acct_name":"0935921",
    "id":9999
  },
  "source":{
    "version":"1.9.3.Final",
    "name":"_7ef6d5bd9f7fac34ca74b91088375d03f4b94d3af0aeef61dec9e9743336d19e",
    "connector":"mysql",
    "ts_ms":1465581029100,
    "snapshot":false,
    "db":"ipaas_test",
    "table":"account_info",
    "server_id":223344,
    "gtid":null,
    "file":"mysql-bin.000005",
    "pos":484,

```



```
    "row":0,
    "thread":7,
    "query":null
  },
  "op":"u",
  "transaction":null,
  "ts_ms":1465581029523
}
```

- **Delete Event:** Delete 操作对应的记录。

```
{
  "before":{
    "account_type":"a",
    "acct_id":9999,
    "acct_name":"0935921",
    "id":9999
  },
  "after":null,
  "op":"d",
  "source":{
    "version":"1.9.3.Final",
    "connector":"mysql",
    "name":"_7ef6d5bd9f7fac34ca74b91088375d03f4b94d3af0aeef61dec9e9743336d19e",
    "ts_ms":1465581902300,
    "snapshot":false,
    "db":"ipaas_test",
    "table":"account_info",
    "server_id":223344,
    "gtid":null,
    "file":"mysql-bin.000005",
    "pos":805,
    "row":0,
    "thread":7,
    "query":null
  },
  "transaction":null,
  "ts_ms":1465581902461
}
```

GTID

当 MySQL Server 由于某些原因不可用时，MySQL Change Data Capture 连接器会停止采集 DML 操作记录，直到 MySQL Server 重新可用，按照 GTID 的配置情况，这里后有两种不同的行为模式：

- 对于打开 GTID 的 MySQL Cluster，连接器会连接到失效 Server 外的另一个 Server 上，找到上一次事务中的 binlog 位置，并开始同步。
- 对于未打开 GTID 的，仅当连接器重连上失效 Server 后，才可以从原来的位置继续同步。

使用示例

1. 新建连接器配置，填写配置参数，单击测试连接，测试连接器配置是否正确。

← 编辑连接配置

×

✔ 连接配置正确

数据库类型

MySQL

数据库Host地址

1. 1. 1. 3

端口号

int

3006

端口号范围：0-65535

用户名

root

账户名称

密码

账户密码

高级配置

▲

安全网关

无

消息确认模式

流运行成功后确认

测试连接

完成

2. 在操作配置中，依次下拉选择数据库、表，选中待同步的数据库和表后，在自动刷新到界面的待选择字段名列表中勾选想要同步的字段，并保存。

配置

备注

通用 ▲

数据库

cdctest2

表

cdctest1

字段

☑ 字段名

操作

☑ id

删除

☑ k

删除

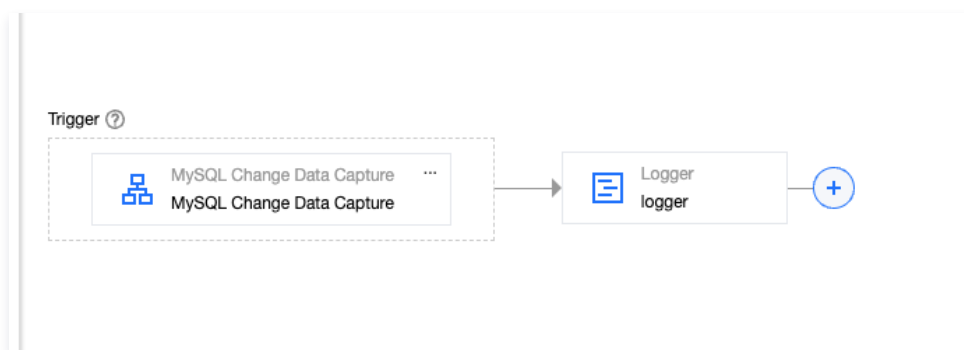
☑ c

删除

☑ pad

删除

3. 在 trigger 节点后增加业务逻辑，此处仅以打印日志作为示例：



管理中心

成员管理

最近更新时间：2024-11-25 09:41:52

操作场景

腾讯云数据连接器是采用 CAM 的账号系统并结合腾讯云数据连接器的单独权限进行权限控制。
腾讯云数据连接器会自动拉取当前主账号下的所有子账号进入成员列表，若要添加/删除成员，需要通过腾讯云数据连接器控制台调整到 CAM 控制台进行操作。主账号成为系统管理员，系统管理员拥有腾讯云数据连接器的最高权限，可设置子账号成为系统管理员或撤销子账号的系统管理员权限。详情请参见 [项目管理 - 项目成员](#)。

操作步骤

添加/删除成员

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**管理中心 > 成员管理**。
2. 在成员管理页面，单击**添加成员/删除**。



3. 在确认弹框中单击**确认前往**，跳转至 CAM 用户列表界面进行操作。



设置/解除系统管理员

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**管理中心 > 成员管理**。
2. 在成员管理页面，单击操作列的**设置为/解除系统管理员**。



查看成员详情

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**管理中心 > 成员管理**。

2. 在成员管理页面，单击目标成员名称，即可查看成员详情。

添加成员

全部成员 仅管理员

成员名称	账号ID	账号类型	权限	操作
CompanyExample	7000000000000000	主账号	系统管理员	--
MemberExample	7000000000000000	子账号	系统管理员	删除 解除系统管理员
MemberExample	7000000000000000	子账号	--	删除 设置为系统管理员

成员详情包括基本信息、授权信息，具体如下：

← 用户详情

基本信息

姓名

7000000000000000

联系手机

--

联系邮箱

--

账号ID

7000000000000000

联系微信

--

创建时间

--

授权信息

所有功能权限

体验项目

HelloWorld

编辑

abc

只读

测试项目

Database

只读

已选择功能权限

体验项目

HelloWorld

编辑

abc

只读

测试项目

Database

只读

项目管理

最近更新时间：2023-06-08 14:48:03

以下视频将为您介绍项目管理的相关概念及操作：

[观看视频](#)

操作场景

集成需求可以通过创建集成应用进行实施，集成应用中可创建多个集成应用，每个集成应用对应一个集成场景。添加集成应用后，可以查看项目详情、修改项目名称、删除项目。

- 项目名称：通过列表展示当前用户有权限的所有项目的名称。
- 默认项目：所有成员默认有默认项目的权限。不支持添加或删除成员，也不支持删除项目，且不支持修改项目名称。
- 项目管理员：最多显示三个项目管理员。
- 描述：用户可以输入文字简单描述项目的内容，方便后续成员进一步区分理解对应项目的内容。
- 项目成员：当前项目包含的成员数量。
- 创建时间：项目首次创建的时间。
- 操作：添加成员、修改项目名称、删除项目。

前提条件

已添加 [集成应用](#)。

操作步骤

添加项目

您可以从项目维度管理集成应用，便于区分业务场景和权限划分。项目支持查看详情、修改名称和删除操作。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击 **管理中心** > **项目管理**。
2. 在项目管理页面，单击**添加项目**。若提示无权限创建项目，可参见 [访问管理示例](#) 分配合理权限。
3. 填写项目名称和描述，单击**确定**完成添加。

添加项目

项目名称 *

请输入项目名称

请输入25位以内的中文、字母、数字、_或-

描述

请输入150字符以内的项目描述

确定

取消

删除项目

当某项目不再使用时，您可以**停止**项目内应用运行后，再删除项目。

⚠ 注意

当项目中存在运行中的应用时，无法删除。且项目删除后数据无法恢复，请谨慎操作。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击 **管理中心** > **项目管理**。

2. 在项目管理页面，单击操作列的删除，并在确认弹框中单击确认删除。

项目管理						帮助			
添加项目						请输入项目名称			
项目名称	项目管理员	描述	项目成员数 (个)	创建时间	操作				
默认项目	-	-	-	-	-				
腾讯问卷	-	-	3	2022-02-11 16:00:30	成员管理 设置管理员 修改 删除				
订单测试	-	-	2	2021-11-19 14:26:25	成员管理 设置管理员 修改 删除				
事件总线2	-	-	4	2021-11-10 21:50:07	成员管理 设置管理员 修改 删除				
事件总线	-	-	2	2021-11-10 10:03:55	成员管理 设置管理员 修改 删除				
企业微信服务	-	-	3	2021-07-30 12:24:23	成员管理 设置管理员 修改 删除				

查看项目

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击 **管理中心 > 项目管理**。

2. 在项目管理页面，单击目标项目名称，进入项目详情页。

项目详情页包含了对应项目下项目集成应用、连接配置和安全网关的列表信息，您可以单击对应名称快速进入编辑页面。

默认项目

描述

-

创建时间

-

深度集成数

4个

连接配置数

7个

安全网关数

1个

深度集成

连接配置

安全网关

自定义错误

应用名称	创建时间	更新时间	运行/调试版本	操作
HelloWorld1	2021-09-07 13:22:13	2021-09-10 16:56:19	-	查看
HelloWorld2	2021-09-10 17:09:20	2021-09-14 09:52:03	运行中: 20210910_1.0(天津)	查看
EB测试汇总	2021-11-10 20:01:24	2021-12-20 17:09:30	-	查看
2	2021-12-15 17:44:55	2021-12-15 17:46:46	-	查看

成员管理

支持进行项目成员管理，可以添加或删除协同开发、运维、发布此项目的成员。可添加或删除的成员列表来源于成员管理。

系统管理员、项目管理员可以添加删除项目中的普通成员，也可以赋予或取消普通成员的项目管理员权限，也可以修改普通用户对该项目下集成应用的只读、编辑、编辑并发布的操作权限。

平台默认创建一个默认项目，平台所有成员都具有默认项目的权限。默认项目不支持添加或删除成员，所以默认项目无项目成员管理栏。

1. 登录 腾讯云数据连接器控制台，在左侧导航栏单击 管理中心 > 项目管理 > 成员管理。

项目管理

添加项目

请输入项目名称

项目名称	项目管理员	描述	项目成员数 (个)	创建时间	操作
默认项目	-	-	-	-	-
腾讯问卷	-	-	3	2022-02-11 16:00:30	成员管理 设置管理员修改删除
订单测试	-	-	2	2021-11-19 14:26:25	成员管理设置管理员修改删除
事件总线2	-	-	4	2021-11-10 21:50:07	成员管理设置管理员修改删除
事件总线	-	-	2	2021-11-10 10:03:55	成员管理设置管理员修改删除
企业微信服务	-	-	3	2021-07-30 12:24:23	成员管理设置管理员修改删除

2. 添加成员

只能添加 CAM 中已有子用户的成员。详细请参见 成员管理。

成员管理

请输入成员名称

共2人

添加成员

基本信息

姓名

180001503407

联系邮箱

180001503407@qq.com

联系微信

180001503407

联系手机

150****8618

账号ID

180001503407

创建时间

-

授权信息

所有功能权限

请输入应用名称

该成员为系统管理员，默认拥有所有权限。

已选择功能权限

该成员为系统管理员，默认拥有所有权限。

确定

3. 设置成员权限

设置成员对应项目内应用，只读、编辑或编辑并发布的权限。

添加成员

成员名称

成员权限

所有功能权限

请输入应用名称

Q

企业微信服务

加解密1

只读 编辑 编辑并发布

3

只读 编辑 编辑并发布

企微成员变更触发集成

只读 编辑 编辑并发布

定时任务

只读 编辑 编辑并发布

HelloWorld

只读 编辑 编辑并发布

已选择功能权限

企业微信服务

加解密1

编辑

3

编辑

企微成员变更触发集成

编辑

定时任务

编辑

HelloWorld

编辑

导入测试

确定

取消

4. 删除成员

删除成员后此用户不再拥有此项目的相应权限。

成员管理

请输入成员名称

Q

共3人

添加成员

8000048881407

已删除

3076090

删除

基本信息

姓名

联系手机

联系邮箱

联系微信

账号ID

创建时间

授权信息

所有功能权限

请输入应用名称

Q

企业微信服务

加解密1

只读 编辑 编辑并发布

3

只读 编辑 编辑并发布

企微成员变更触发集成

只读 编辑 编辑并发布

定时任务

只读 编辑 编辑并发布

已选择功能权限

企业微信服务

加解密1

编辑

3

编辑

企微成员变更触发集成

编辑

定时任务

编辑

HelloWorld

编辑

确定

设置管理员

支持为已有项目设置项目管理员，选择项目成员并将其设置为项目管理员后，此成员将拥有对应项目的最高权限，只有项目成员才能设置成为项目管理员。

登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击 **管理中心 > 项目管理 > 设置管理员**。

项目管理

添加项目

请输入项目名称

项目名称	项目管理员	描述	项目成员数 (个)	创建时间	操作
默认项目	-	-	-	-	-
腾讯问卷	-	-	3	2022-02-11 16:00:30	成员管理 设置管理员 修改 删除
订单测试	-	-	2	2021-11-19 14:26:25	成员管理 设置管理员 修改 删除
事件总线2	-	-	4	2021-11-10 21:50:07	成员管理 设置管理员 修改 删除
事件总线	-	-	2	2021-11-10 10:03:55	成员管理 设置管理员 修改 删除
企业微信服务	-	-	3	2021-07-30 12:24:23	成员管理 设置管理员 修改 删除

修改项目

支持修改项目，可对已存在的项目的项目名称及描述进行修改。

项目管理

添加项目

请输入项目名称

项目名称	项目管理员	描述	项目成员数 (个)	创建时间	操作
默认项目	-	-	-	-	-
腾讯问卷	-	-	3	2022-02-11 16:00:30	成员管理 设置管理员 修改 删除
订单测试	-	-	2	2021-11-19 14:26:25	成员管理 设置管理员 修改 删除
事件总线2	-	-	4	2021-11-10 21:50:07	成员管理 设置管理员 修改 删除
事件总线	-	-	2	2021-11-10 10:03:55	成员管理 设置管理员 修改 删除
企业微信服务	-	-	3	2021-07-30 12:24:23	成员管理 设置管理员 修改 删除

修改项目名称及描述

项目名称

企业微信服务

请输入25位以内的中文、字母、数字、_或-

描述

请输入150字符以内的项目描述

确定 取消

环境管理

最近更新时间：2023-06-08 14:48:03

以下视频将为您介绍环境管理的相关概念及操作：

[观看视频](#)

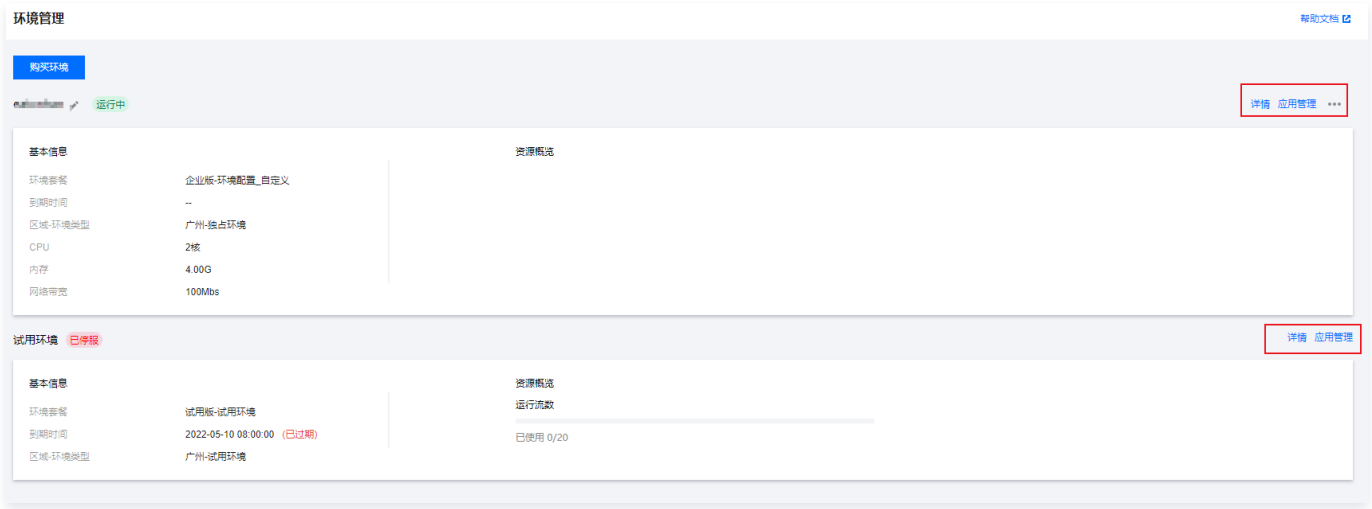
操作场景

环境管理向用户提供一个便捷的环境和运行时环境查看、管理的路径，您可以一站式的查看部署环境的配置、使用情况以及运维信息。

环境管理页面

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**管理中心 > 环境管理**。

2. 环境列表页面展示各个环境的基本信息，如下图所示：



- **环境名称：**展示当前用户有权限的所有集群的名称。

○ **状态：**展示每个环境的瞬时状态，此处的展示会随着环境状态的变化而变化，目前暂不支持控制台操作创建、删除和扩容等操作，目前平台支持以下几种集群状态（后端执行对应操作，前端展示状态会更新），其中不同状态环境的可用性是有区别的：

环境状态	可用性
创建中	不支持发布
运行中	支持发布
伸缩中	支持发布
删除中	不支持发布
不可用	不支持发布
创建失败	不支持发布
已到期	不支持发布

- **区域-环境类型：**展示独立及共享环境的环境列表。当前平台支持两种不同的环境类型，不同的环境类型描述了不同环境的部署特点和资源能力。

○ 共享环境：所有用户共享的高可用的腾讯云数据连接器的运行环境，可用于快速发布应用。

○ 独立环境：基于数据和资源的安全性考虑，腾讯云数据连接器也提供资源和数据隔离性更强的部署模式，即独立环境。用户可以独占腾讯云数据连接器的运行环境，达到计算资源和数据的隔离，能够以较低成本，快速提升隔离级别。目前独立环境已开放创建，需要使用该服务可以单击[购买环境](#)进行购买。

○ **环境套餐：**企业版（企业版独立环境才显示此字段）。

○ **到期时间：**对应环境使用的到期时间。

○ **资源：**CPU、内存、网络带宽的资源只有企业版环境显示购买时或升级后的配置信息，共享环境不展示。

○ **资源概览：**展示企业版环境的实际资源使用情况，共享环境不展示。

○ **操作：**目前支持应用管理、运行概况及配置信息等操作。

环境管理相关操作

购买环境

⚠ 注意

此购买的独立环境仅用于应用集成，不可用于 API 管理。

共享环境适用于新用户测试产品的基本功能，而独立环境（企业版）更适合作为线上业务的部署环境。

独立环境基于数据和资源的安全性考虑，提供了资源和数据隔离性更强的部署模式。用户可以独占腾讯云数据连接器的运行环境，达到计算资源和数据的隔离，能够以较低成本，快速提升隔离级别。适合作为线上业务的部署环境。

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏单击**管理中心** > **环境管理**，单击**购买环境**。



2. 进入配置选型界面，根据业务的需求选择对应的环境配置即可。若担心过高的配置造成资源浪费，可先选择基础环境配置，后续**变更配置**即可。更多购买相关内容可参见 [购买指南](#)。

购买企业版环境 权益及资源对比

×

购买时长

1个月

3个月

6个月

1年

环境配置

环境配置A 适用约 2 个应用系统集成	CPU: 4核 安全网关: 4个	内存: 8G 日志存储: 50GB	网络带宽: 200Mbps 可运行流数: 80条	13200/月
环境配置B 适用约 4 个应用系统集成	CPU: 8核 安全网关: 8个	内存: 16G 日志存储: 50GB	网络带宽: 400Mbps 可运行流数: 160条	26400/月
环境配置C 适用约 8 个应用系统集成	CPU: 16核 安全网关: 16个	内存: 32G 日志存储: 50GB	网络带宽: 800Mbps 可运行流数: 320条	52800/月
环境配置D 适用约 16 个应用系统集成	CPU: 32核 安全网关: 16个	内存: 64G 日志存储: 50GB	网络带宽: 1600Mbps 可运行流数: 640条	105600/月

若不满足需求, 请 [联系我们](#)

续费策略

☐ 自动续费 [①](#)

地域

广州

协议条款

☐ 我已阅读并同意 [《腾讯云数据连接器服务等级协议》](#)

费用

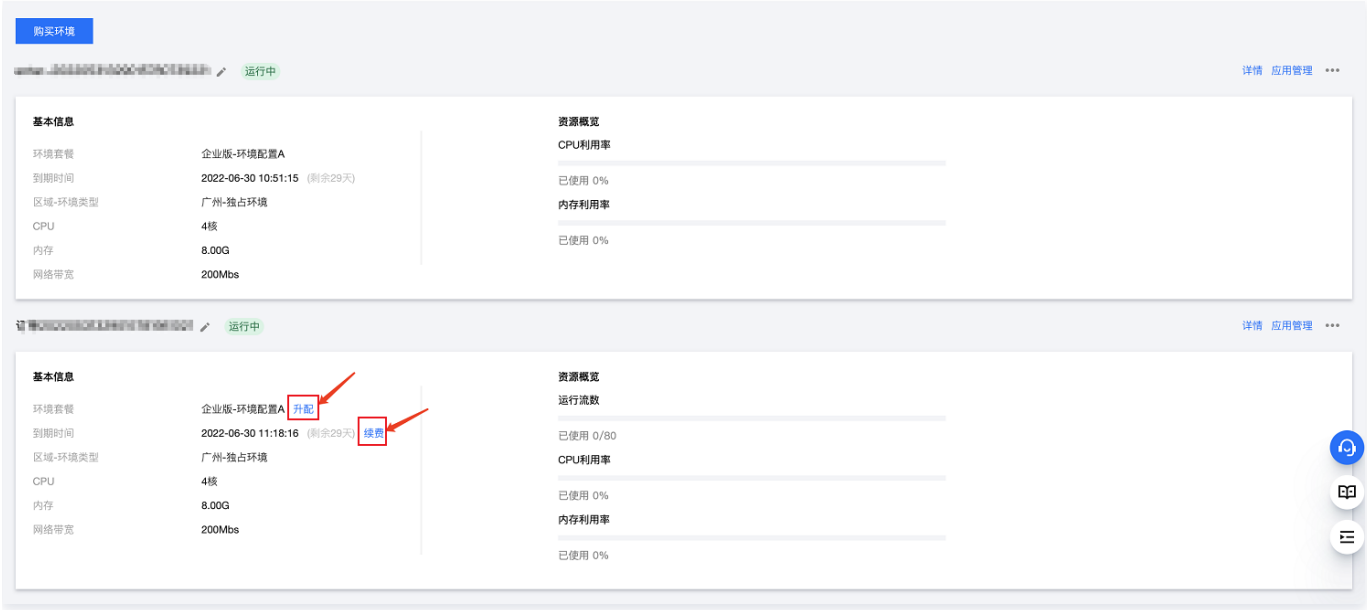
13200元

立即购买

取消

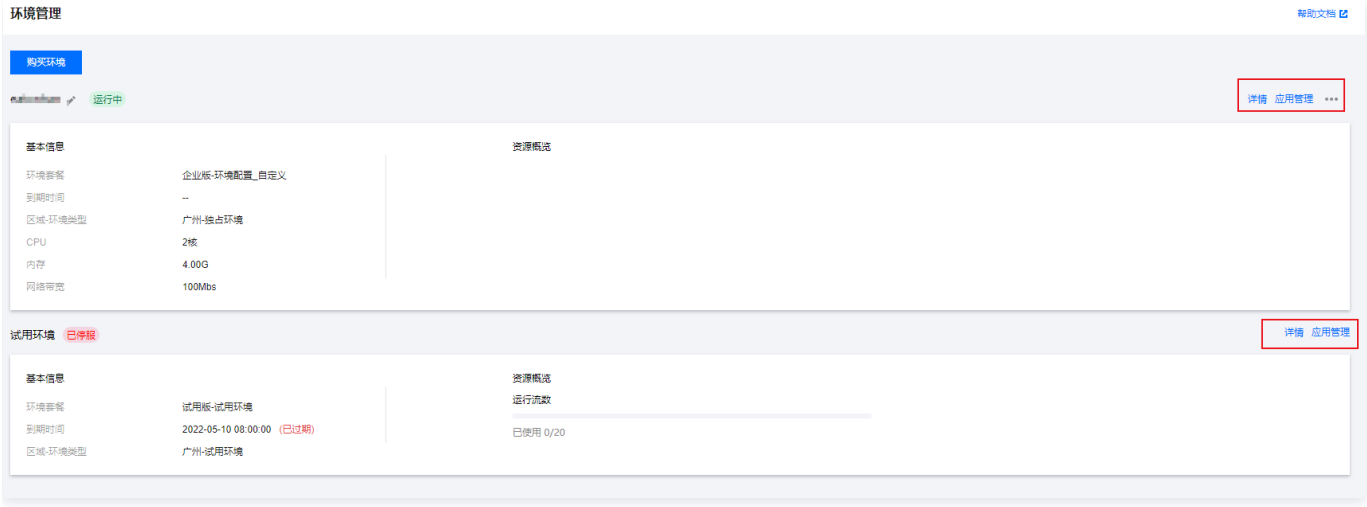
续费和升配

您可以在环境管理中对购买的企业版环境进行续费和升配。
登录到 [环境管理](#)，单击续费或升配，即可操作。



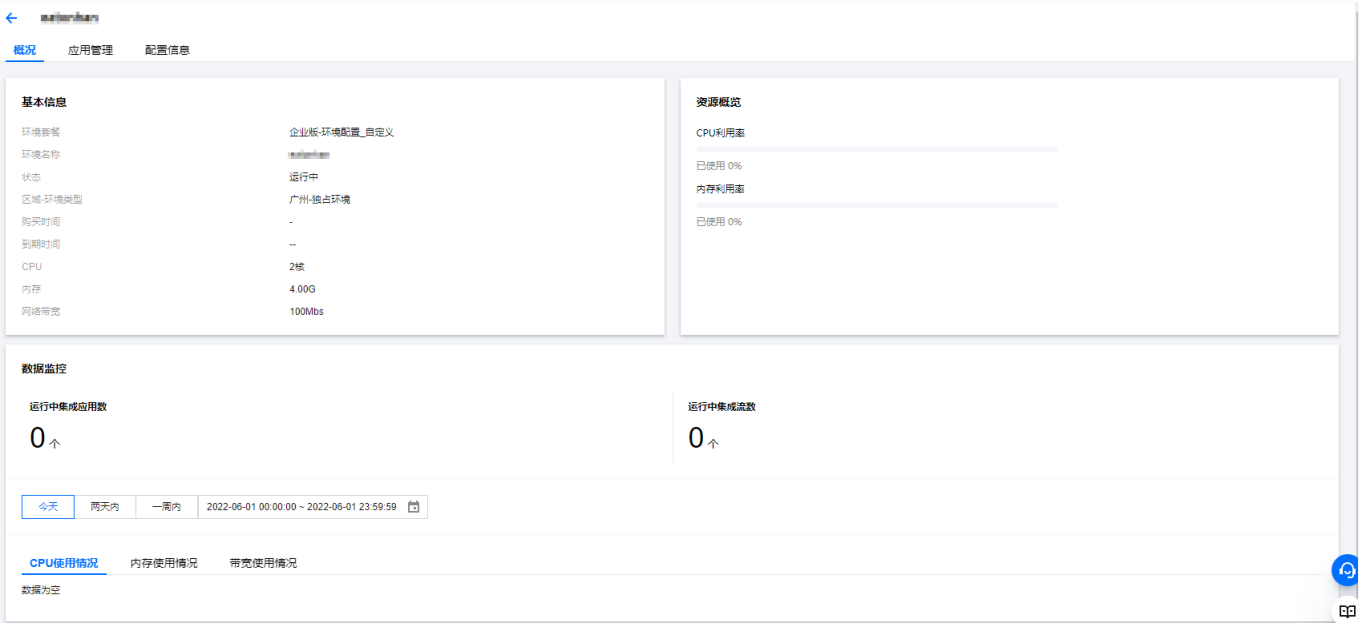
查看环境详情

单击对应环境详情，可查看环境详情，包括：概况、应用管理、配置详情（独立环境展示）等信息。



概况

在环境详情页的概况页签，对环境名称、状态、区域-环境类型、运行中集成应用数等相关信息做了集中展示，但共享环境和独立环境所展示的字段略有不同。如下图（红框标注为企业版环境特有展示字段，试用版环境不展示）：



- **共享环境：**共享环境（试用版）的概况会展示该环境的基本信息、运行区域、运行状态等信息，同时会展示该环境中运行中的集成应用数以及运行中的集成流数。
- **独立环境：**独立环境（企业版）的概况与测试环境类似，但是除了展示该环境的基本信息、运行区域、运行状态等信息，同时会展示该环境中运行中的集成应用数以及运行中的集成流数之外，还会展示当前环境的使用、运行状况（如CPU内存等的使用情况）。

应用管理

在环境详情页的**应用管理**页签，或从**环境管理>应用管理**，进入**应用管理**页签。此页面集中展示当前环境下的应用名称、应用状态、项目名称、部署时间、运行/调试版本及操作等内容，方便客户进一步查看和管理指定环境中的相关应用。

应用名称	状态	项目名称	部署时间	运行/调试版本	操作
http_测试	已停止	测试-demo	2022-05-10 15:24:38	20220510_3.0	开始 移除
http_测试	已停止	测试-demo	2022-05-10 15:17:10	20220510_2.0	开始 移除
http_测试	已停止	测试-demo	2022-05-10 15:16:14	20220510_1.0	开始 移除
异常集成测试	运行中	测试-demo	2022-04-15 15:44:05	20220310_5.0	停止
企业demo	运行中	测试-demo	2022-03-18 10:34:50	20220318_15.0	停止
异常集成测试	已停止	测试-demo	2022-03-10 08:54:31	20220310_4.0	开始 移除
异常集成测试	已停止	测试-demo	2022-03-10 00:22:53	20220310_3.0	开始 移除
API测试	已停止	测试-demo	2022-03-09 23:58:44	20220307_2.0	开始 移除
异常集成测试	已停止	测试-demo	2022-03-09 23:57:16	20220309_2.0	开始 移除
wywmawm_demo	运行中	医疗行业	2022-03-03 10:45:44	20220303_1.0	停止

应用管理的相关操作：

- **开始：**对环境内处于**已停止**状态的应用进行启动操作。
- **停止：**对环境内处于**运行中**状态的应用进行停止操作。

- 移除：对环境处于已停止状态的应用进行移除该环境的操作，移除后，将不支持日志查询。

应用管理

概况

应用管理

配置信息

应用名称	状态	项目名称	部署时间	运行调试版本	操作
http_demo	已停止	http_demo	2022-05-10 15:24:38	20220510_3.0	开始 移除
http_demo	已停止	http_demo	2022-05-10 15:17:10	20220510_2.0	开始 移除
http_demo	已停止	http_demo	2022-05-10 15:16:14	20220510_1.0	开始 移除
轻量级测试	运行中	轻量级测试	2022-04-15 15:44:05	20220310_5.0	停止
企业版测试	运行中	企业版测试	2022-03-18 10:34:50	20220318_15.0	停止
轻量级测试	已停止	轻量级测试	2022-03-10 08:54:31	20220310_4.0	开始 移除
轻量级测试	已停止	轻量级测试	2022-03-10 00:22:53	20220310_3.0	开始 移除
API测试	已停止	API测试-demo	2022-03-09 23:58:44	20220307_2.0	开始 移除
轻量级测试	已停止	轻量级测试	2022-03-09 23:57:16	20220309_2.0	开始 移除
轻量级测试_demo	运行中	轻量级测试	2022-05-30 18:15:45	20220303_1.0	停止

共 15 条

10 条 / 页

1 / 2 页

配置详情

注意

独立环境才展示配置详情页面，共享环境目前是共享环境使用，后台默认配置相关参数不可修改。

在环境详情页的应用管理页签，集中展示了环境配置（对应环境下的使用限制）、Dataway 配置、组件配置等参数配置，用户可根据使用场景修改对应上限值，修改后单击保存即可，恢复默认值单击重新配置。如下图：

应用管理

概况

应用管理

配置信息

环境配置

使用限制

每秒最大请求数

300

项目每秒最大请求数

200

流每秒最大请求数

200

最大并发量

30

项目最大并发量

20

流最大并发量

20

执行引擎

最大消息大小

10

MB

内存最大消息大小

1

MB

当前配置需重启才能生效

流执行超时时间

720

分钟

Dataway配置

超时时间

60000

毫秒

当前配置需重启才能生效

组件配置

DB

Fetch Size

1000

当前配置需重启才能生效

保存

重新配置

- 环境配置
 - 使用限制
 - 每秒最大请求数：每秒最多可触发请求多少次。
 - 项目每秒最大请求数：单个项目每秒最多可触发请求多少次。
 - 流每秒最大请求数：单个流每秒最多可触发请求多少次。
 - 最大并发量：系统可支持的最大并发量。

- 项目最大并发量：项目最大并发量。
- 流最大并发量：单个流的最大并发量。
- 执行引擎
 - 最大消息大小：Message 在内存中的大小限制。
 - 流执行超时时间：流执行超时时间。
 - 内存最大消息大小：超过此大小的消息将保存至 COS 进行处理。
- DataWay 配置
 - 超时时间：DataWay 超时时间，配置后需重启对应集成应用才能生效。
- 组件配置
 - DB
 - Fetch Size：每次最多从数据库查询多少条数据，配置后需重启对应集成应用才能生效。

审计日志

最近更新时间：2023-06-08 14:48:03

操作场景

审计日志支持查看一段时间内的历史操作事件，包括基本信息、操作信息和事件详情，通常可用于合规审计、项目变更溯源和安全分析等使用场景。

操作步骤

步骤1：查询审计日志

控制台默认展示近1天的审计日志，用户可设置筛选项来查看对应审计日志。

!

说明

目前控制台支持检索近一个月的审计日志。若有检索更早期日志的需求，请 [提交工单](#) 联系我们。

审计日志

近3天

全部用户

全部项目

全部操作模块

全部操作类型

全部操作结果

请输入操作事件关键词

搜索

重置

日志数量 37条

下载日志

操作时间	用户名称(ID)	操作结果	项目名称	操作模块	操作类型	操作事件	操作
2022-03-24 16:56:28	腾讯云用户ID	成功	默认项目	API管理	新建	新建API服务(我是api服务名称)	查看详情
2022-03-24 16:55:52	腾讯云用户ID	成功	测试项目	项目管理	新建	新建项目(测试项目)	查看详情
2022-03-23 17:51:05	腾讯云用户ID	成功	测试项目	API管理	新建	新建API服务(呜呜呜呜)	查看详情
2022-03-22 17:48:30	腾讯云用户ID	成功	测试项目	项目管理	新建	新建项目(全部)	查看详情
2022-03-22 11:15:46	腾讯云用户ID	失败	默认项目	深度集成	编辑	对应用(空白应用1)中连接器(tencent-docs)进行版本修改	查看详情
2022-03-22 11:15:45	腾讯云用户ID	成功	默认项目	深度集成	编辑	对应用(空白应用1)中连接器(wenjuan)进行版本修改	查看详情
2022-03-22 11:15:45	腾讯云用户ID	成功	默认项目	深度集成	编辑	对应用(空白应用1)新增集成流(NewFlow)	查看详情
2022-03-22 11:15:44	腾讯云用户ID	成功	默认项目	深度集成	新建	添加应用(空白应用1)	查看详情
2022-03-22 11:01:39	腾讯云用户ID	失败	默认项目	深度集成	编辑	对应用(0322_110132)中连接器(tencent-docs)进行版本修改	查看详情
2022-03-22 11:01:39	腾讯云用户ID	成功	默认项目	深度集成	编辑	对应用(0322_110132)中连接器(wenjuan)进行版本修改	查看详情
2022-03-22 11:01:38	腾讯云用户ID	成功	默认项目	深度集成	编辑	对应用(0322_110132)新增集成流(NewFlow)	查看详情
2022-03-22 11:01:37	腾讯云用户ID	成功	默认项目	深度集成	新建	添加应用(0322_110132)	查看详情

1. 登录 [腾讯云数据连接器控制台](#)，在左侧导航栏中选择**管理中心 > 审计日志**。
2. 设置日志筛选项。筛选项包括：

字段	说明
操作时间	支持按操作时间进行搜索，预定义近5分钟、近15分钟、近1小时、近6小时、近1天、近3天、近7天、近30天、今天、昨天、前天、本周、上周，以及自定义时间区间的设置。
用户名称	支持按用户名称以及账号ID 进行搜索，可用于查看指定用户的审计日志。
项目名称	支持按项目名称进行搜索，可用于查看属于指定项目的审计日志。
操作模块	支持按操作模块进行搜索，包括集成应用、连接管理、安全网关、运行监控、通用存储、集成资源、项目管理、API 管理、API 用户中心等功能模块。
操作类型	支持按操作类型进行搜索，包括新建、编辑、删除、发布、停止。
操作结果	支持按操作结果进行搜索，包括成功、失败。
自定义搜索框	支持按用户输入的操作事件关键词进行搜索，通常用于异常追溯场景。

3. 查看日志。显示对应筛选条件的审计日志，按就近原则进行排序。日志列表内容包括：

字段	说明
操作时间	操作事件发生的时间。
用户名称	操作事件对应的执行者用户名称以及账号ID。
操作结果	操作事件是否执行完成。若失败，则代表该操作未执行完成，支持查看错误信息。
项目名称	操作事件所属的项目。
操作模块	操作事件对应的功能模块。
操作类型	操作事件对应的抽象操作类型。
操作事件	操作事件的内容摘要。
操作	查看详情。支持查看基本信息、操作信息和事件详情（见 步骤2 ）。

4. 重置日志筛选项。

当前筛选查询完毕后，可通过单击**重置**快速恢复默认展示。

审计日志

近7天

全部项目

连接管理、通用存储、... 3项

新建、编辑、删除

全部操作结果

请输入操作事件关键词

搜索

重置

日志数量 51条								↓ 下载日志
操作时间	用户名(ID)	操作结果	项目名称	操作模块	操作类型	操作事件	操作	
2022-04-02 17:50:27	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:09:28	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:03:49	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:03:41	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:03:12	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:03:02	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:02:59	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:02:39	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Database组件优化演示)中集成流(database)进行编辑	查看详情	
2022-04-02 16:02:11	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Table组件优化演示)中集成流(table测试-插入)进行编辑	查看详情	
2022-04-02 16:02:06	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Table组件优化演示)中集成流(table测试-插入)进行编辑	查看详情	
2022-04-02 15:09:48	admin@192.168.1.100	成功	默认项目	深度集成	编辑	对应用(Table组件优化演示)中集成流(table测试-插入)进行编辑	查看详情	

步骤2：查看日志详情

审计日志							日志详情																																																																																					
近3天全部用户全部项目 全部操作模块全部操作类型全部操作结果请输入操作事件关键词搜索重置							基本信息 操作时间2022-03-24 17:31:02 事件ID67hDu...DAP_5NsT 用户名称 账号ID 源IP地址																																																																																					
日志数量 41条 <table><tr><th>操作时间</th><th>用户名称(ID)</th><th>操作结果</th><th>项目名称</th><th>操作模块</th><th>操作类型</th><th>操作事件</th></tr><tr><td>2022-03-24 17:31:02</td><td></td><td>失败</td><td>默认项目</td><td>API管理</td><td>新建</td><td>新建API服务(我</td></tr><tr><td>2022-03-24 17:24:09</td><td></td><td>成功</td><td>默认项目</td><td>API管理</td><td>新建</td><td>新建API服务(我</td></tr><tr><td>2022-03-24 17:23:57</td><td></td><td>失败</td><td>默认项目</td><td>API管理</td><td>新建</td><td>新建API服务(我</td></tr><tr><td>2022-03-24 17:23:21</td><td></td><td>成功</td><td>默认项目</td><td>API管理</td><td>新建</td><td>新建API服务(我</td></tr><tr><td>2022-03-24 16:56:28</td><td></td><td>成功</td><td>默认项目</td><td>API管理</td><td>新建</td><td>新建API服务(我</td></tr><tr><td>2022-03-24 16:55:52</td><td></td><td>成功</td><td>测试项目</td><td>项目管理</td><td>新建</td><td>新建项目(测试项</td></tr><tr><td>2022-03-23 17:51:05</td><td></td><td>成功</td><td></td><td>API管理</td><td>新建</td><td>新建API服务(鸣</td></tr><tr><td>2022-03-22 17:48:30</td><td></td><td>成功</td><td></td><td>项目管理</td><td>新建</td><td>新建项目(全部</td></tr><tr><td>2022-03-22 11:15:46</td><td></td><td>失败</td><td>默认项目</td><td>深度集成</td><td>编辑</td><td>对应用(空白应用1</td></tr><tr><td>2022-03-22 11:15:45</td><td></td><td>成功</td><td>默认项目</td><td>深度集成</td><td>编辑</td><td>对应用(空白应用1</td></tr><tr><td>2022-03-22 11:15:45</td><td></td><td>成功</td><td>默认项目</td><td>深度集成</td><td>编辑</td><td>对应用(空白应用1</td></tr></table>							操作时间	用户名称(ID)	操作结果	项目名称	操作模块	操作类型	操作事件	2022-03-24 17:31:02		失败	默认项目	API管理	新建	新建API服务(我	2022-03-24 17:24:09		成功	默认项目	API管理	新建	新建API服务(我	2022-03-24 17:23:57		失败	默认项目	API管理	新建	新建API服务(我	2022-03-24 17:23:21		成功	默认项目	API管理	新建	新建API服务(我	2022-03-24 16:56:28		成功	默认项目	API管理	新建	新建API服务(我	2022-03-24 16:55:52		成功	测试项目	项目管理	新建	新建项目(测试项	2022-03-23 17:51:05		成功		API管理	新建	新建API服务(鸣	2022-03-22 17:48:30		成功		项目管理	新建	新建项目(全部	2022-03-22 11:15:46		失败	默认项目	深度集成	编辑	对应用(空白应用1	2022-03-22 11:15:45		成功	默认项目	深度集成	编辑	对应用(空白应用1	2022-03-22 11:15:45		成功	默认项目	深度集成	编辑	对应用(空白应用1	操作信息 项目名称默认项目 操作模块API管理 操作类型新建 操作结果失败 错误信息API服务版本小于当前最新服务版本 (9.11.1) 操作事件新建API服务(我是api服务名称)	
操作时间	用户名称(ID)	操作结果	项目名称	操作模块	操作类型	操作事件																																																																																						
2022-03-24 17:31:02		失败	默认项目	API管理	新建	新建API服务(我																																																																																						
2022-03-24 17:24:09		成功	默认项目	API管理	新建	新建API服务(我																																																																																						
2022-03-24 17:23:57		失败	默认项目	API管理	新建	新建API服务(我																																																																																						
2022-03-24 17:23:21		成功	默认项目	API管理	新建	新建API服务(我																																																																																						
2022-03-24 16:56:28		成功	默认项目	API管理	新建	新建API服务(我																																																																																						
2022-03-24 16:55:52		成功	测试项目	项目管理	新建	新建项目(测试项																																																																																						
2022-03-23 17:51:05		成功		API管理	新建	新建API服务(鸣																																																																																						
2022-03-22 17:48:30		成功		项目管理	新建	新建项目(全部																																																																																						
2022-03-22 11:15:46		失败	默认项目	深度集成	编辑	对应用(空白应用1																																																																																						
2022-03-22 11:15:45		成功	默认项目	深度集成	编辑	对应用(空白应用1																																																																																						
2022-03-22 11:15:45		成功	默认项目	深度集成	编辑	对应用(空白应用1																																																																																						
							事件详情 事件详情 <pre>{ "API服务名称": "我是api服务名称", "协议": 1, "描述": "我是描述", "标签": "我是标签", "版本": "1.2.3"}</pre>																																																																																					

类型	字段	说明
基本信息	事件 ID	每条审计日志的唯一标识。
	源 IP 地址	操作事件对应用户的源 IP 地址。
操作信息	错误信息	当操作结果为失败时，会显示对应错误信息
时间详情	时间详情	显示当前操作事件对应的请求参数。例如：操作事件为“新建 API 服务{我是 API 服务名称}”，则此处会记录 API 服务名称、协议、描述、标签、版本等信息。

平台提供下载日志的功能。支持将当前列表的审计日志导出为 .xlsx 或 .csv 文件，以便后续进一步分析或整理。详细字段可参见 [日志列表](#)。

审计日志

近1天

全部用户

全部项目

全部操作模块

全部操作类型

全部操作结果

请输入操作事件关键词

搜索

重置

日志数量 6条

操作时间	用户名(ID)	操作结果	项目名称	操作模块	操作类型	操作事件	操作
2022-03-25 17:08:39	大刘(测试账号)	成功	test	深度集成	编辑	对应用(问卷信息采集_0325_170825)中集成流(问卷信息采集)进行编辑	导出为csv文件 导出为xlsx文件
2022-03-25 17:08:29	大刘(测试账号)	成功	test	深度集成	编辑	对应用(问卷信息采集_0325_170825)中途接驳(腾讯文档)进行版本修改	查看详情
2022-03-25 17:08:29	大刘(测试账号)	成功	test	深度集成	编辑	对应用(问卷信息采集_0325_170825)中途接驳(腾讯问卷-Cauth)进行版本修改	查看详情
2022-03-25 17:08:29	大刘(测试账号)	成功	test	深度集成	编辑	对应用(问卷信息采集_0325_170825)新增集成流(问卷信息采集)	查看详情
2022-03-25 17:08:28	大刘(测试账号)	成功	test	深度集成	新建	添加应用(问卷信息采集_0325_170825)	查看详情
2022-03-25 17:07:44	大刘(测试账号)	成功	test	项目管理	新建	新建项目(test)	查看详情

DataWay 表达式简介

最近更新时间：2024-11-07 11:41:52

以下视频将为您介绍 DataWay 表达式的应用场景、相关概念及 Debug 调试：

[观看视频](#)

DataWay 是腾讯云数据连接器（APP Flow）中用于对流运行数据进行自定义转换与处理的表达式脚本语言，集成在腾讯云数据连接器运行时服务中，是提供腾讯云数据连接器可扩展性的关键能力。

腾讯云数据连接器的许多内置组件和连接器中都可提供基于 DataWay 表达式的自定义能力，可以用于对腾讯云数据连接器事件进行动态处理。例如在 Set-Variable 组件中，可以通过 DataWay 表达式来动态地设置变量的值；在 Transform 组件中，可以充分利用 DataWay 的灵活语法进行复杂的数据处理与表达式运算，以最终生成期望的产出结果，用于下游组件的处理。

您可以通过在腾讯云数据连接器内置组件配置中直接使用 DataWay 表达式。

- 要直接开始使用 DataWay 表达式，请参见 [DataWay 快速入门](#)。
- 已经有腾讯云数据连接器开发经验，要了解更多关于 DataWay 特性，请参见 [DataWay 开发指南](#)。
- 要查看 DataWay 中的常用函数，请参见 [函数参考](#)。
- 如果您有其他问题，请参见 [常见问题](#)。

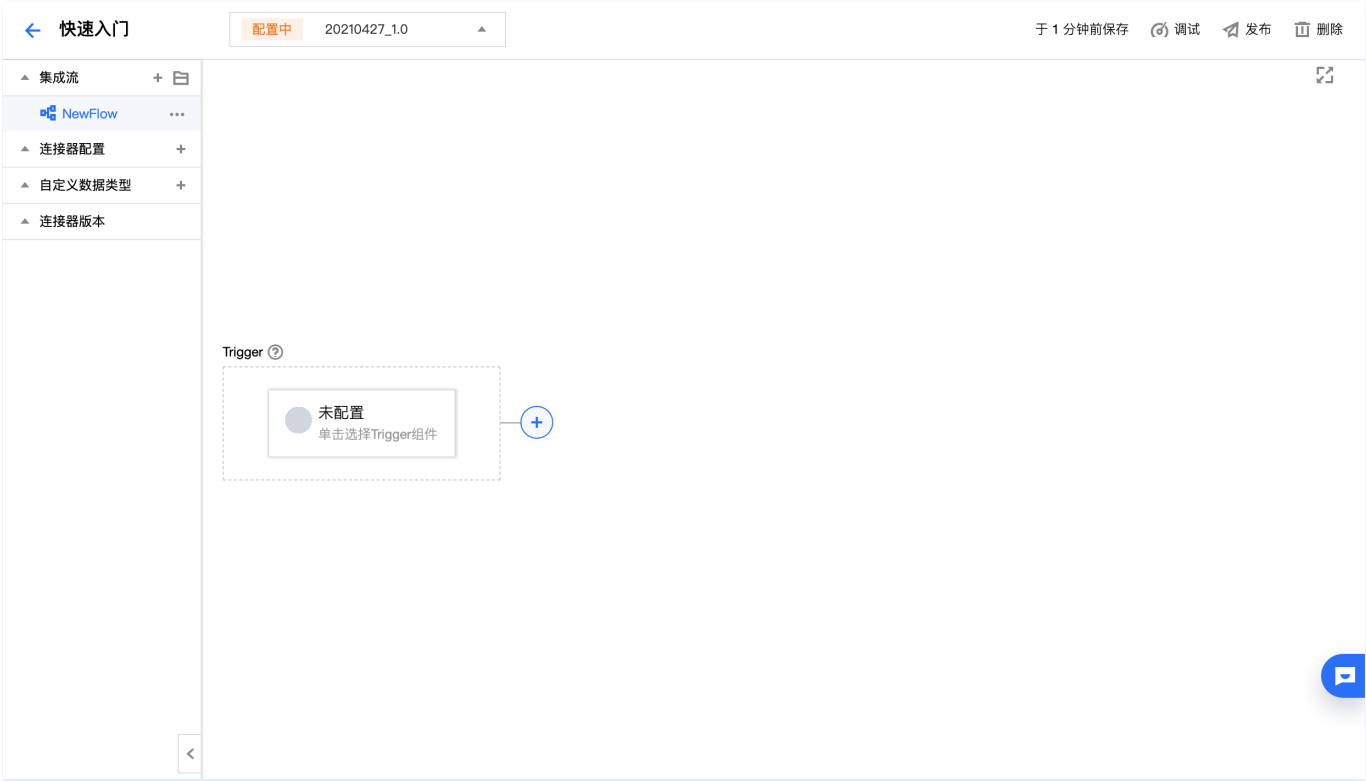
快速入门

最近更新时间：2023-06-08 14:48:04

本节将介绍如何使用 Dataway 表达式来辅助进行集成流设计。

前期准备

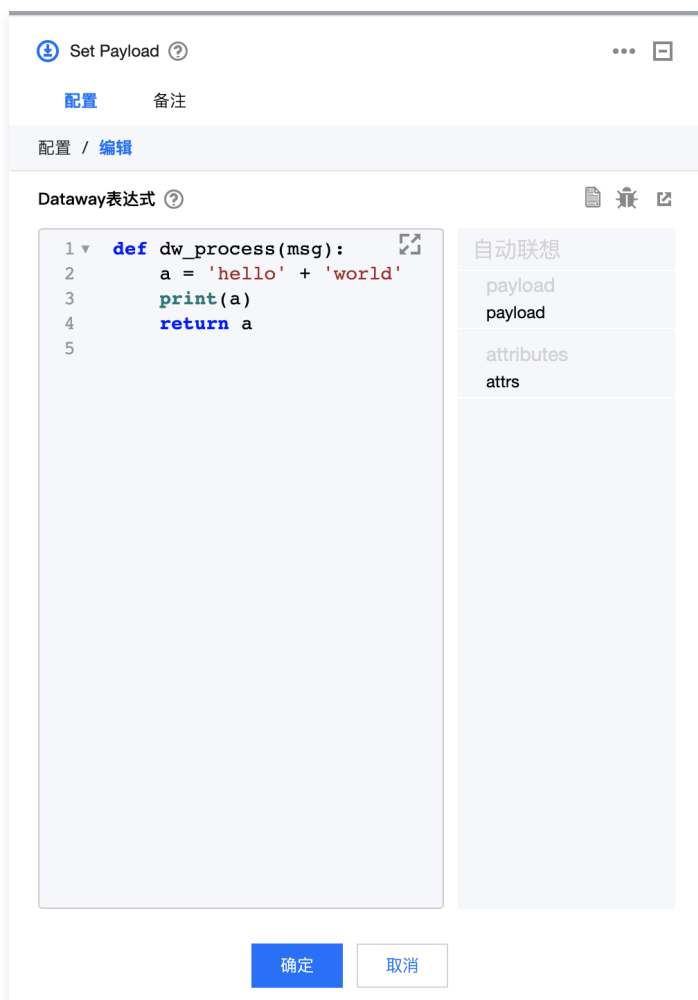
要使用 Dataway 表达式，需要完成腾讯云数据连接器平台的账号申请。申请成功后，新建一个应用并创建一条集成流。



使用 Dataway 表达式

本小节以一个简单的字符串连接为示例。

1. 在流中新建一个 Set Payload 组件，并展开组件配置。在"值"配置项中，单击右侧编辑框，进入 Dataway IDE。



2. 进入 Dataway 编辑框，输入 Dataway 表达式。输入时会实时进行语法检查，若出错则会有对应提示。

```
def dw_process(msg):
    return 'hello' + 'world'
```

- 完整的 DataWay 脚本符合语法定义的 Python3 代码段，其中包含入口函数定义 `def dw_process(msg)`。

- DataWay 基于 Python3.7 语法进行实现，同时内置了多个第三方模块，如 time、json、math等，使用时直接引用模块名即可。



3. 验证 Dataway 运行结果。在通过语法检查并单击**确定**保存表达式之前，即可对 Dataway 脚本的正确性进行验证。在编辑框右上角单击“Debug”图标，在弹出的对话框直接单击**开始测试**。

Set Payload ?

配置
备注

配置 / 编辑

Dataway表达式 ?

1
2
3
4
5

def dw_process(msg):
a = 'hello' + 'world'
print(a)
return a

自动联想

payload
payload
attributes
attrs

确定
取消

请模拟表达式的输入数据

payload 值
string
请输入字符串

attribute 值
变量名
变量值
操作

+添加变量

variable 值
变量名
变量值
操作

+添加变量

开始测试
取消

4. 测试结束后，Dataway 编辑框的下方会有输出结果的展示，可以看到 Dataway 脚本的运行结果为 `hello world`，表示脚本的返回结果符合预期。同时可以切换到日志项，查看 print 的输出结果。

Set Payload ?

...

[-]

配置

备注

配置 / 编辑

Dataway表达式 ?

1

def dw_process(msg):

2

a = 'hello' + 'world'

3

print(a)

4

return a

5

自动联想

payload

payload

attributes

attrs

输出 ✓

错误

日志

"helloworld"

确定

取消

5. 单击**确定**，即完成 Dataway 脚本的保存。

更多资料

关于 Dataway 的使用，更多资料请参见 [DataWay 开发指南](#)。

函数参考

最近更新时间：2024-12-11 16:27:42

内置函数

内置函数				
abs()	enumerate()	isinstance()	object()	slice()
all()	filter()	iter()	ord()	sorted()
any()	float()	len()	pow()	str()
bool()	getattr()	list()	print()	sum()
bytearray()	hasattr()	map()	range()	tuple()
bytes()	hash()	max()	reversed()	type()
chr()	id()	min()	round()	zip()
dict()	int()	next()	set()	

目前 DataWay 中支持的内置函数如下：

- abs ()：求数值绝对值
- all ()：判断序列（集合、列表、元组、dict）中所有元素是否满足给定条件
- any ()：判断集合中是否存在元素满足给定条件
- bool ()：构造布尔值
- bytearray ()：构造字节数组
- bytes ()：构造空字节
- chr ()：0~256的整数对应的 ASCII 码
- dict ()：创建字典
- enumerate ()：将一个可遍历的数据对象组合列出数据和数据下标，一般用于 for 循环中
- filter ()：集合过滤，例如：list(filter(lambda x:x>=100, [1,3,4,100,102])) -> [100,102]
- float ()：构造浮点数
- getattr ()：求一个对象的属性值
- hasattr ()：判断一个对象是否有某个属性
- hash ()：求哈希值
- id ()：求对象的唯一标识
- int ()：构造整数
- isinstance ()：判断对象是否属于某种类型
- iter ()：生成一个迭代器
- len ()：获取集合元素个数
- list ()：构造列表
- map ()：根据函数对指定序列做映射，例如：list(map(lambda x: x * 2, [1, 2, 3, 4, 5])) -> [2, 4, 6, 8, 10]
- max ()：获取数值最大值
- min ()：数值最小值
- next ()：返回迭代器的下一个项目，和 iter() 一起使用
- object ()：返回空对象
- ord ()：单个 ASCII 码字符的整数值
- pow ()：求指数
- print ()：DataWay 调试时可以打印相关信息（仅在 DataWay 表达式编辑时使用 Debug 功能生效）
- range ()：创建可迭代对象，例如：list(range(5)) -> [0, 1, 2, 3, 4]
- reversed ()：创建反转的迭代器，例如：list(reversed('abcdefg')) -> ['g', 'f', 'e', 'd', 'c', 'b', 'a']
- round ()：截取数值的整数部分

- `set()`：创建一个集合
- `slice()`：设置截取元素的切片
- `sorted()`：排序
- `str()`：构造字符串
- `sum()`：数值求和
- `tuple()`：构造元组
- `type()`：返回对象类型
- `zip()`：打包可迭代对象中元素成多个 tuple。例如：`list(zip([1,2,3], [4,5,6])) -> [(1, 4), (2, 5), (3, 6)]`

内置第三方模块

- `__time__`，用于时间处理的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。
 - 目前 DataWay 中支持的库函数/类型如下：
 - `altzone`：当前时区相对于 UTC 时区的延迟偏移，单位为秒。
 - `asctime`：将一个 `struct_time` 转换为时间字符串。
 - `ctime`：将一个时间戳转换为时间字符串。
 - `mktime()`：将一个 `struct_time` 转换为时间戳。
 - `strftime()`：将一个 `struct_time` 进行格式化。
 - `strptime()`：按照给定的格式解析事件字符串，并返回一个结构化的 `struct_time` 对象。
 - `timezone`：当前时区。
 - `tzname`：当前时区名称。
 - `time()`：当前时间。
 - `localtime`：将一个时间戳转换为当前时区的本地时间，返回 `struct_time` 类型对象。
- `__json__`，用于处理 json 数据的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。
 - 目前 DataWay 中支持的 json 模块函数：
 - `dumps()`：将 json 对象编码为 json 字符串。
 - `loads()`：将一个 json 串解析为 Python 对象。
- `__math__`，用于数学运算的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。
 - 目前 DataWay 中支持的 math 模块函数：
 - `math.ceil(x)`：返回 x 的上限，即大于或者等于 x 的最小整数。如果 x 不是一个浮点数，则委托 `x.ceil()`，返回整数。
 - `math.floor(x)`：返回 x 的向下取整，即小于或等于 x 的最大整数。如果 x 不是浮点数，则委托 `x.floor()`，返回整数。
 - `math.fabs(x)`：返回 x 的绝对值。
 - `math.pow(x, y)`：返回 x 的 y 次幂。
 - `math.sqrt(x)`：返回 x 的平方根。
 - 支持的常量：
 - `math.pi`：数学常数 $\pi = 3.141592\dots$ ，精确到可用精度。
 - `math.e`：数学常数 $e = 2.718281\dots$ ，精确到可用精度。
 - `math.inf`：浮点正无穷大。（对于负无穷大，使用 `-math.inf`。）相当于 `float('inf')` 的输出。
 - `math.nan`：浮点“非数字”（NaN）值，相当于 `float('nan')` 的输出。
- `__base64__`：用于 base64 编解码的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。支持的函数有：
 - `base64.b64encode(s)`：对 bytes 类型参数 base64 编码，返回编码后的 bytes。
 - `base64.b64decode(s)`：对 bytes/str 类型参数 base64 解码，返回解码后的 bytes。
- `__hmac__`：用于 hmac 加解密的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。支持的函数有：
 - `hmac.new(key)`：返回一个新的 hmac 对象，key 是一个指定密钥的 bytes 或 bytearray 对象。
- `__random__`：用于随机数生成的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。支持的函数有：
 - `random.randint(a, b)`：返回随机整数 N 满足 $a \leq N \leq b$ 。
- `__hashlib__`：用于生成哈希值的库，可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中，可以直接引用。支持的函数/属性有：
 - `hashlib.sha256()`：创建一个 SHA-256 hash 对象。
 - `hashlib.md5()`：创建一个 MD5 的 hash 对象。
 - `hashlib.sha1()`：创建一个 SHA1 的 hash 对象。

- `__datetime__`: 用于时间日期处理的库, 可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `__datetime.date__`: 一个理想化的简单型日期, 它假设当今的公历在过去和未来永远有效。属性有: `year`、`month` 和 `day`。
 - `__datetime.time__`: 一个理想化的时间, 它独立于任何特定的日期, 假设每天一共有 246060 秒。属性有: `hour`、`minute`、`second`、`microsecond` 和 `tzinfo`。
- `__datetime.datetime__`: 日期和时间的结合。属性有: `year`、`month`、`day`、`hour`、`minute`、`second`、`microsecond` 和 `tzinfo`。
- `__datetime.timedelta__`: 表示两个 `date` 对象或者 `time` 对象, 或者 `datetime` 对象之间的时间间隔, 精确到微秒。
- `__datetime.timezone__`: 表示相对于世界标准时间 (UTC) 的偏移量。
- `__datetime.tzinfo__`: 描述时区信息对象, 用来给 `datetime` 和 `time` 类提供自定义的时间调整概念, 例如处理时区和/或夏令时。
- `__decimal__`: 用于定点数处理的库, 可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `decimal.Decimal`: 表示十进制浮点数对象。
- `__socket__`: Python 中 tcp 套接字的底层实现, 可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `socket.htonl()`: 把32位正整数从网络序转换成主机字节序。
 - `socket.ntohl()`: 将 32 位正整数从网络字节序转换为主机字节序。
- `__pycryptodome__`: 专用的加密工具三方库, 可参见 [pycryptodome 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `__Crypto.Util.Padding__`: 该模块为添加和删除数据中的标准填充提供了最小的支持, 提供 `pad()`、`unpad()` 方法。
 - `__Crypto.Cipher.AES__`: AES加密模块。它具有16字节的固定数据块大小, 密钥可以是128、192或256位长。提供 `new()` 方法。
- **struct**: 用于打包二进制文件的库, 可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `struct.pack (format, v1, v2, ...)`: 返回一个 `bytes` 对象, 其中包含根据格式字符串 `format` 打包的值 `v1`, `v2`, ... 参数个数必须与格式字符串所要求的值完全匹配。
 - `struct.unpack (format, buffer)`: 根据格式字符串 `format` 从缓冲区 `buffer` 解包, 返回一个 `tuple`。
- **urllib**: 用于 URL 处理的库, 可参考 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `urllib.parse.urlparse()`: 获取 url 参数, 将 url 解析成一个包含6个字符串的 `tuple`, 分别为: 协议, 位置, 路径, 参数, 查询, 片段识别。
 - `urllib.parse.unquote()`: 将编码的 url 进行解码。
- **csv**: 用于 CSV 文件读写的库, 可参见 [Python 官方文档](#)。已内置在 DataWay 的处理上下文中, 可以直接引用。支持的函数/属性有:
 - `csv.reader()`: 创建一个 `reader` 对象, 将逐行遍历 CSV 文件对象。

开发指南

DataWay 语言基础概念

最近更新时间：2025-04-22 11:40:11

DataWay 语言是一门在腾讯云数据连接器中用于对数据进行自定义转换与处理的表达式脚本语言。使用 DataWay 可以编写出强大和复杂的数据转换程序，在此之前需要先了解 DataWay 中的核心概念和功能。

脚本结构

- 完整的 DataWay 脚本符合语法定义的 Python 3代码段，其中包含入口函数定义 `def dw_process(msg)`，例如：

```
def dw_process(msg):
    sq = func(3)
    val = {
        'square': sq,
        'data': msg.payload['realData'] + 1
    }
    return Entity.from_value(val, mime_type='application/json')
def func(x):
    return x*x
```

- `dw_process` 入口函数仅接受一个参数 `msg`，该参数代表当前表达式需要处理的腾讯云数据连接器消息，`dw_process` 的返回值即是表达式的返回值。
- 内置的 `Entity.from_value` 函数用于为构造 `Entity` 类型的返回值，可以指定序列化参数，例如：`mime_type`、`encoding` 等。
- 在 `Set Payload` 组件中输入上述表达式，假设该组件的输入消息为 `json` 结构的数据 `{"realData": 123}`，经过 `DataWay` 表达式的计算，得到的输出结果如下：

```
{
  "square": 9,
  "data": 124
}
```

基本语法说明

DataWay 基于 Python 3.7语法进行实现，本小节将对 DataWay 的基本语法进行说明。

关键字

在 DataWay 中，支持的关键字如下表所示。关键字作为 DataWay 中的保留字，不会被当成任何标识符名称。

关键字名称	说明
True	布尔类型，True 表示真，相对于 False
False	布尔类型，False 表示假，相对于 True
None	空值类型
and	逻辑 “与”
or	逻辑 “或”
not	逻辑 “非”
as	自定义命名
assert	断言，用来测试表达式
break	终止循环语句
continue	跳过本次循环
def	函数定义

if/else/elif	判断语句
for	循环语句
global	全局变量声明
in	判断是否包含在其中
is	判断两个变量指向是否一致
lambda	匿名函数，可以用一行实现一个函数
nonlocal	在嵌套函数中声明，可以修改外部定义的变量
pass	空语句，用于占位
raise	抛出异常
return	函数返回

行和缩进

DataWay 使用缩进来标识代码块，不同的缩进行数代表不同的代码层级，同一层级的缩进行数需保持一致。

运算符

DataWay 支持常见的运算符：算数运算符、比较运算符、赋值运算符、逻辑运算符、位运算符等。下表列出了常见的运算符。其中，假设变量 a 为5、b 为3。

运算符名称	说明	示例
=	赋值	<code>c = 3</code>
+	加	<code>a + b = 8</code>
-	减	<code>a - b = 2</code>
*	乘	<code>a * b = 15</code>
/	除	<code>15 / a = b</code>
%	取模，返回除法的余数	<code>16 % b = 1</code>
**	幂	<code>a ** b = 125</code>
//	向下取整	<code>a // b = 1</code>
+=	加法赋值	<code>c += a</code> 等效于 <code>c = c + a</code>
-=	减法赋值	<code>c -= a</code> 等效于 <code>c = c - a</code>
*=	乘法赋值	<code>c *= a</code> 等效于 <code>c = c * a</code>
/=	除法赋值	<code>c /= a</code> 等效于 <code>c = c / a</code>
==	是否相等	<code>a == b</code> 返回 False
!=	是否不等	<code>a != b</code> 返回 True
>	大于比较	<code>a > b</code> 返回 True
<	小于比较	<code>a < b</code> 返回 False
>=	大于等于比较	<code>a >= b</code> 返回 True
<=	小于等于比较	<code>a <= b</code> 返回 False
&	按位与	<code>a & b = 1</code> (0101 & 0011 = 0001)
	按位或	<code>a b = 7</code> (0101 & 0011 = 0111)
^	按位异或	<code>a ^ b = 6</code> (0101 ^ 0011 = 0110)

~	按位取反	~a = -6
<<	左移运算符	a << 3 = 20 (0000 0101 << 3 = 0001 0100)
>>	右移运算符	a >> 1 = 2 (0101 >> 1 = 0010)

条件及循环控制语句

- DataWay 通过 if/elif/else 语句来进行条件控制。示例如下：通过判断 a 的值，返回不同的字符串：

```
def dw_process(msg):
    a = 100
    if a < 10:
        return 'a is lower than 10'
    elif a <= 100 and a >= 10:
        return 'a is between 10 and 100'
    else:
        return 'a is bigger than 100'
```

DataWay 表达式的运行结果为：a is between 10 and 100

- DataWay 通过 for 循环进行循环控制。示例如下：通过 for 循环，得到 a 中元素的乘积：

```
def dw_process(msg):
    a = [1, 2, 3, 4]
    num = 1
    for i in a:
        num *= i
    return num
```

DataWay 表达式的运行结果为：24

定义函数

- 在 DataWay 中，可以使用 def 关键词定义函数，后接函数名和参数名列表，以冒号 “:” 作为定义函数行的结尾，下一行默认缩进；最终以 return 语句结束函数，如果不带 return 则相当于返回 None。
- 定义一个函数后，可以在另一个函数中调用执行。在 DataWay 中，默认的入口函数 dw_process 函数无需手工声明。如果想自定义函数，直接在 dw_process 入口函数下方定义即可。如下示例，定义一个函数 test() 用于对列表元素求和，并在 dw_process() 函数中调用，最终使用 return 语句返回结果。

```
def dw_process(msg):
    a = [1, 2, 3, 4]
    return add_list(a)
def add_list(alist):
    sum = 0
    for i in reversed(alist):
        sum += i
    return sum
```

最终的输出结果为：10

模块调用

DataWay 内置多个第三方模块，例如：time、json、math、base64、hmac、random、hashlib、Crypto、socket、struct、decimal 和 datetime 等，使用时直接引用模块名即可，无需使用 import 关键字。具体的函数说明可参见 [DataWay 函数参考](#)。具体示例如下，接收一个 json 类型字符串，转换成一个 dict 字典：

```
def dw_process(msg):
    jsonStr = '{"a": 1, "b": 2, "c": 3}'
    jsonDict = json.loads(jsonStr) # 转换成一个dict
    num = 1
    for k, v in jsonDict.items(): # 对dict进行遍历
```

```
num += math.pow(v, 2)
return num
```

最终的输入结果为: 15.0

注释

DataWay 单行注释以 # 开头，多行注释则可以用多个 # 号，或者 ''' 和 """。举例如下，执行下面代码：

```
# Dataway 注释
'''
Dataway 注释
Dataway 注释
'''

"""
Dataway 注释
Dataway 注释
"""

def dw_process(msg):
    return 'Dataway Hello World!'
```

输出结果为：

```
Dataway Hello World!
```

⚠ 注意：

DataWay 提供语法检查功能，在编写代码时会进行实时语法检查，并给出错误提示。详细的 Python 语法说明可以参见 [Python 官方文档](#)。

dw_process 入口函数

- dw_process 是 DataWay 的主入口函数，其作用相当于 C/C++ 语言中的 main 函数。
- dw_process 仅接受一个类型为 [Message](#) 的参数，而其返回值就是该 DataWay 表达式最终的输出值。
- 作为腾讯云数据连接器数据处理流程的一个环节，dw_process 函数的返回值目前支持的类型有：str/None/bool/float/int/list/dict/Entity/MultiMap/FormDataParts/Message 等。
- 关于 DataWay 中数据类型及返回值的详细介绍，可参见 [DataWay 数据类型系统](#)。

DataWay 数据类型系统

类型名	说明	是否 DataWay 特有类型	举例
str	字符串，即 Python 原生的字符串 str	否	"abc"
None	Python 中的空值 None	否	None
bool	布尔值，即 Python 原生布尔值 bool	否	True/False
float	浮点数，即 Python 原生浮点数 float	否	123.123
int	整数，即 Python 原生整型 int	否	123
bytes	字节数组，即 Python 字节数组类型 bytes	否	b'this_is_a_bytes'
set	集合，即 Python 集合类型 set	否	{1,2,3}
list	列表，序列类型容器，即 Python 原生 list 类型	否	[1,2,3]
dict	字典，kv 类型容器，即 Python 原生 dict 类型	否	{1:1, 'key': 'value'}
Entity	即腾讯云数据连接器中的实体数据，用于代表一个二进制对象，在 DataWay 中以 Entity 类型进行访问，包括 blob、	是	HTTP-listener 构造消息中的 payload，如 msg.payload

	mime_type、encoding 等信息		
MultiMap	多值 map，类似于 xml 而与 dict 不同，该类型可以支持重复的 key	是	application/www-form-urlencoded 格式的数据解析之后得到的对象
FormDataParts	数组+列表的数据结构，类似于 Python 中的 orderDict 结构	是	multipart/form-data 格式的数据解析后得到的对象
Message	即腾讯云数据连接器中的消息，在 dataWay 中以 Message 进行访问	是	dw_process 入口函数中的 msg 参数

⚠ 注意：

- 上述类型可以在 DataWay 表达式中使用，但 dw_process 函数的返回值的类型为其中的 str/None/ bool/float/int/list/dict/Entity/MultiMap/FormDataParts/Message 之一。
- 需要注意的是，如果 DataWay 表达式输出的值会作为集成流的最终返回结果，则支持的返回值类型还会受到相应连接器组件的限制。如在以 HTTP listener 组件作为第一个组件的流中，其最终的 payload 也需要是一个 Entity 类型。

Message 类型及预定义属性

Message 类型是 DataWay 用于表示一条腾讯云数据连接器消息的数据类型，其中包含 payload、vars、attrs 等属性，称之为**预定义属性（Predefined Properties）**。这些属性是由系统根据当前运行信息及处理的消息生成的，用于在 DataWay 中通过程序化的方式获取上下文信息。

目前 Message 中包含的属性及其说明如下：

属性	作用	属性类型	属性说明
msg.vars	当前消息上下文中的变量	dict 类型，键为 str 类型，代表变量名，值为允许的任意类型，代表变量值	vars 会在消息处理的所有环节共享，因此可用于在不同的处理节点之间进行数据的传递
msg.payload	当前消息的载荷数据	任意类型	payload 是一条消息对象的负载数据，一般是由上一个组件通过 Set-Payload 或者 Transform 组件生成的，其可能的数据类型即是上文中列举的 DataWay 中支持的返回值类型。HTTP listener 组件会根据用户发送的原始消息来构造 payload 的内容，因此 listener 组件处理之后，payload 都是 Entity 类型，除非在下游通过 set-payload 或者 transform 组件对 payload 进行重新赋值
msg.attrs	当前消息的属性数据，如消息来源、消息的头部信息等	dict 类型，键为 str，代表属性名，值为任意类型，代表属性值	如果 trigger 组件为 HTTP listener，则请求的 headers 将会被设置到 msg.attrs 中
msg.id	当前消息的唯一标识 id	str 类型	经过一个逻辑组件，msg.id 可能会变化
msg.seq_id	当前消息的序列号	str 类型	消息在流中流转时，msg.seq_id 保持不变
msg.error	当前处理上下文中的错误信息	dict 类型，键为 str，代表错误属性名；值为 str，代表属性值	包含的内容有：msg.error['code']：错误类型；msg.error['desc']：错误描述字符串

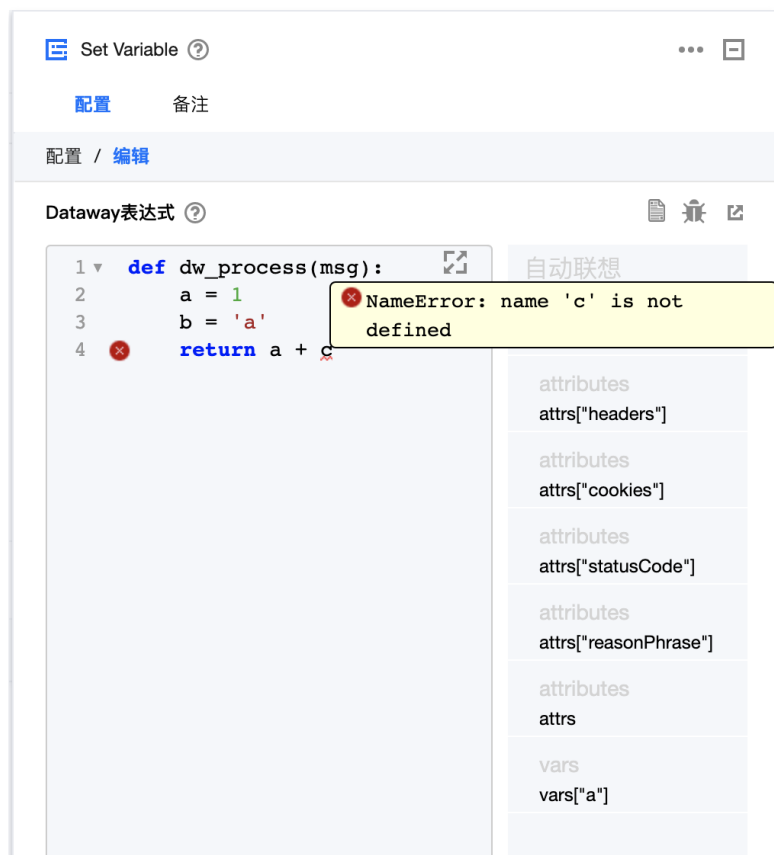
DataWay IDE 使用

在腾讯云数据连接器系统新建一个 Set Variable 组件，单击变量值文本框进入 DataWay 脚本编辑器，该编辑器提供语法检查、格式化、脚本调试、自动联想、代码高亮等类 IDE 功能。



语法检查

在 DataWay IDE 中，能够实时对 DataWay 表达式进行语法检查，并通过强提示显示在 IDE 左侧和出错代码底部。当鼠标移动到错误提醒处，会有详细的错误信息说明。用户可以根据语法提示来对 DataWay 脚本进行修改，只有语法检查通过的代码才能够保存成功。



格式化

在 Dataway IDE 中，提供有格式化功能按钮。用户可以单击右上角格式化按钮，一键对 DataWay 代码格式化，以使代码更加简洁规范。



单击右上角格式化按钮后，格式化后的代码如下图：



脚本调试

在 DataWay IDE 中，提供有 Debug 调试功能按钮。用户可以单击右上角“Debug”图标，在线对 DataWay 脚本进行调试。详细的使用方法可见 [DataWay 脚本调试](#)。



自动联想

DataWay IDE 能够自动对流中已配置的 payload、vars、attrs 中的变量名进行联想，并展示在 IDE 右侧栏。例如：在 Set Variable 组件前已经创建了一个 Set Payload 和一个 Set Variable 组件，则右侧栏会展示出来，单击对应的联想值会自动加入到左侧编辑框中。

Set Variable ?

配置

备注

配置 / 编辑

Dataway表达式 ?

1

def dw_process(msg):

2

return msg.vars['a']

3

|

自动联想

payload

payload

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["statusCode"]

attributes

attrs["reasonPhrase"]

attributes

attrs

vars

vars["a"]

代码高亮

Dataway IDE 默认对 Dataway 代码进行高亮和括号自动匹配。

版权所有：腾讯云计算（北京）有限责任公司

第695 共732页

使用 Entity 对象

最近更新时间：2023-09-13 14:26:21

类型

- 在 DataWay 中，用 Entity 类型来表示腾讯云数据连接器中的实体数据，表示二进制数据的封装对象，其主要组成部分包括 blob、mime_type 以及 encoding。
 - blob：原始的二进制数据。
 - mime_type：表示二进制数据的内容格式，例如：application/json、application/www-form-urlencoded、multipart/form-data 等。
 - encoding：表示二进制数据的字符编码格式，例如：utf8、gbk 等。
- 可以通过如下方式访问 Entity 中的内容：
 - entity['^blob']：获取该消息对象的负载数据，返回 bytes 类型的对象。
 - entity['^mime_type']：获取该消息对象的 mime_type，返回 str 对象。
 - entity['^encoding']：获取该消息对象的 encoding，返回 str 对象。
 - entity['^value']：根据 mime_type 和 encoding，对负载数据 blob 进行反序列化，并返回其结果，该类型为 [DataWay 类型系统](#) 中定义的类型之一。
 - entity[xxxx]：根据 mimetype/encoding 对 message 的内容进行反序列化，并获取其中指定 key 的值，逻辑上相当于 obj['^value']['xxx']。
 - entity.get(attr, default=None)：根据 mimetype/encoding 对 message 的内容进行反序列化，并获取其中指定 key 的值，如果获取不到则返回默认值 None，逻辑上相当于 obj['^value'].get(attr, default=None)。

选择器

对于 Entity 类型的变量，如预定义属性 msg.payload，DataWay 支持通过选择器 (selector) 的方式进行快速访问，支持的操作类型如下：

下标类型	描述	举例
数字	访问当前数组的第 i 个元素	msg.payload[0]
以 '^' 开头的字符串	获取元信息，例如：^mime_type、^encoding、^raw（原始二进制）、^value（值）	msg.payload['^mime_type']
普通字符（字母、数字、下划线、横杠、点）	普通字符的 key，按名称的方式获取当前元素的子元素，如果有多个同名的，只返回第一个	msg.payload['list']

下面将对这几种选择器类型举例说明。假设输入消息的 payload 值为一个 Entity 类型，其代表的负载数据 blob 反序列化后为一个 json 数组 `[{"a1":1}, {"b1":1, "b2":2, "b3":3}, {"c1":[1,2,3]}]`，对应的 mime_type 为 application/json，编码为默认的 utf-8。

使用数字下标获取数据

使用数字下标可以获取输入 payload 中的元素。如下所示，Dataway 表达式为：

```
def dw_process(msg):  
    return msg.payload[1]
```

则表达式的输出为 dict 类型的结果：`{"b1":1, "b2":2, "b3":3}`

使用 '^' 符号获取元数据

要获取 Entity 类型的 payload 元数据，可以使用 '^' 符号。如下所示，DataWay 表达式为：

```
def dw_process(msg):  
    return {  
        "mimeType": msg.payload["^mime_type"],  
        "encoding": msg.payload["^encoding"],  
        "blob": msg.payload["^blob"],  
        "value": msg.payload["^value"],  
    }
```

则表达式的输出为 dict 类型的结果：

```
{
    "mimeType" : "application/json",
    "encoding": "utf-8",
    "blob": b'[{ "a1":1},{ "b1":1,"b2":2,"b3":3},{ "c1":[1,2,3]}]',
    "value": [{ "a1":1},{ "b1":1,"b2":2,"b3":3},{ "c1":[1,2,3]}]
}
```

使用普通字符获取元素

我们假设输入消息的 payload 值仍为 Entity 类型，其代表的负载数据 blob 反序列化后变成 `{ "a1":1, "b1":1, "b2":2, "b3":3, "c1":[1,2,3]}`，对应的 mime_type 仍为 application/json，编码为默认的 utf-8。如下所示，DataWay 表达式为：

```
def dw_process(msg):
    return {
        "a1" : msg.payload["a1"],
        "b2": msg.payload["b2"],
        "c1": msg.payload['c1'],
    }
```

则表达式的输出为 dict 类型的结果：

```
{
    "a1" : 1,
    "b2": 2,
    "c1": [1,2,3]
}
```

⚠ 注意

在 DataWay 表达式中，字符串可以用 `' '` 或 `" "` 的方式表示，其作用一致。

类型对象构造

Entity.from_value 函数

用于将值类型 data 封装为 Entity 类型，并返回。如下所示：

```
Entity.from_value(data, mime_type=None, encoding="utf-8")
```

在 from_value 函数内部，会先根据给定的 mime_type 和 encoding 尝试对 data 进行序列化得到 bytes 类型的数据，然后再封装为 Entity 类型进行返回。

- mime_type 参数必传，且目前支持的 mime_type 参数有 text/plain、application/json、application/x-www-form-urlencoded、application/csv、application/xml 和 multipart/form-data 六种。
- encoding 参数则支持任意合法的编码类型，若为空则默认为 utf-8 编码。

Entity.from_bytes 函数

用于将 str 或者 bytes 类型数据封装为 Entity 类型，并返回。如下所示：

```
Entity.from_bytes(data, mime_type=None, encoding="utf-8")
```

from_bytes 函数中的 mime_type 和 encoding 参数校验规则与 from_value 函数相似，不同的是不会对 mime_type 的参数值进行限制，可以为任意的 mime_type 类型。

- 如果传递给 from_bytes 函数的 data 参数类型为 bytes 类型，该函数会直接返回一个以 data、mime_type、encoding 作为参数构造的 Entity 对象。
- 如果传递的 data 是 str 数据，则会尝试根据 encoding 参数将其编码为 bytes 类型，并构造出 Entity 对象。

使用对象限制

Entity 对象在本质上是一个对二进制数据的封装对象，为方便使用，也提供例如：Entity.get(attr, default=None) 等对象方法以及基于下标的选择器语法（详情请参见 [Entity 选择器](#)）等快速访问功能。

在使用这些功能时，需要注意的是，有些特殊操作下系统会尝试先对 Entity 中的二进制数据进行解析，如果解析失败则会导致运行时错误。这些特殊操作包括：

- 用 `Entity['^value']` 来获取解析后的结构化结果。
- 用 `Entity.get('attrName', default=None)` 或 `Entity['attrName']` 来获取结构化结果中的某个属性值。
- 用 `Entity['*key']` 来执行选择器语法（只在 `mime_type` 为 `application/x-www-form-urlencoded` 的 Entity 中使用）。

在不符合条件的 Entity 对象上执行上述特殊操作，将会导致运行时错误。

MimeType 支持

最近更新时间：2024-12-11 16:22:02

Dataway 使用 Entity 类型可支持多种不同的数据类型，例如：json、csv、xml 等。在 Entity.from_value 或 Entity.from_bytes 函数中指定对应的 mime_type 和 encoding 值，可以得到一个封装不同数据类型的 Entity 对象。通过使用 Entity 选择器，可以读取解析后的结构数据。不同的 mime_type 在 Entity 中有不同的数据格式，对应关系如下：

mime_type	数据格式
application/json	JSON 格式
text/json	JSON 格式
application/x-www-form-urlencoded	HTTP 表单格式
text/plain	文本格式
application/xml	XML 格式
text/xml	XML 格式
application/csv	CSV 表单格式
multipart/form-data	HTTP FORM DATA 表单格式
其他 mime_type	其他格式

不同的数据格式有不同的编码规则、数据结构以及特定的 Entity 选择器语法。本节将对这些不同的数据格式分别进行说明。

JSON 格式

JSON 格式的数据代表 mimeType 为 application/json或text/json的Entity中数据序列化后得到的类型。

- 使用 Entity.from_bytes 方法，则 Dataway 对输入的 str/bytes 类型最终解析成一个 dict 类型。
- 使用 Entity.from_value 构造方法，支持 list/dict/MultiMap 等多种输入类型，并最终解析成一个 dict 类型数据结构。

以下将通过两个示例来说明如何使用 JSON 格式数据。

构造 JSON 格式的 Entity

本示例将展示如何使用 Entity 的构造函数构造一个 JSON 格式的 Entity，并使用 Entity 选择器获取 Entity 的属性和数据。

输入

DataWay 的运行环境依赖于组件的运行，假定在 Set-Payload 组件前已经有一个 Transform 组件，对流的运行消息 msg 的 payload 进行了设置。msg.payload 为一个 dict 类型对象，内部结构如下：

```
{
  "name": "zhangsan",
  "age": 10,
  "male": True,
  "brothers": ["lisi", "zhaowu"]
}
```

DataWay 脚本

以下 DataWay 脚本使用 from_value 函数将 dict 数据类型的 msg.payload 转换成 Entity 对象，然后用选择器获取 Entity 的元数据和元素，并返回。

```
def dw_process(msg):
    entity = Entity.from_value(msg.payload, mime_type='application/json', encoding='utf-8')
    return {
        'blob': entity['^blob'],
        'mimeType': entity['^mime_type'],
        'name': entity['name'],
        'brother': entity['brothers'][0],
        'male': entity['^value']['age'],
        'other': entity.get('other', 'other_default')
```

```
}
```

输出

DataWay 的脚本输出为一个 dict，其结果如下：

```
{
  "blob": b'{"name":"zhangsan","age":10,"male":true,"brothers":["lisi","zhaowu"]}',
  "mimeType": "application/json",
  "name": "zhangsan",
  "brother": "lisi",
  "male": 10,
  "other": "other_default"
}
```

JSON 结构使用

本示例将对复杂的 JSON 结构进行解析，并运行在 Set-Payload 组件中。

输入

Dataway 的运行环境依赖于组件的运行，假定在 Set-Payload 组件前已经有一个 Transform 组件，对流的运行消息 msg 的 payload 进行设置。msg.payload 为一个 Entity 类型对象，内部结构如下：

```
{
  'mime_type': 'application/json',
  'encoding': 'utf-8',
  'blob': b'{"name":"zhangsan","age":10,"male":true,"brothers":["lisi","zhaowu"]}'
}
```

DataWay 脚本

以下 DataWay 脚本将对 msg 进行处理，获取对应的元素，最终返回一个 Entity 对象。

```
def dw_process(msg):
    val = {
        "k0": msg.payload['^blob'].decode(msg.payload['^encoding']),
        "k1": {
            "str": "str",
            "list": [123, "abc", 12.34],
            "dict": {
                "k1": "v1"
            }
        },
        "k2": math.sqrt(10),
        "k3": json.dumps({"a": 1, "b": [1, "d"]}),
        "k4": time.strftime('%Y-%m-%d %H:%M:%S'),
        "k5": "hello" + "world",
        "k6": ["12", 3] + [[1, 2], 4.5],
        "k8": msg.payload['^mime_type'],
    }
    if val['k8'] == 'application/json':
        val['k7'] = msg.payload.get('gbk')
        val['k9'] = time.time()
    return Entity.from_value(val, mime_type='application/json', encoding='utf-8')
```

输出

DataWay 脚本的输出结果为一个 Entity 类型对象，为方便说明，我们将 Entity 中的 blob 进行反序列化后赋值给 value 属性。

```
{
```

```
"mime_type": "application/json",
"encoding": "utf-8",
"value": {
  "k0": "{\\"name\\":\\"zhangsan\\",\\"age\\":10,\\"male\\":true,\\"brothers\\":[\\"lisi\\",\\"zhaowu\\"]}",
  "k1": {
    "str": "str",
    "list": [
      123,
      "abc",
      12.34
    ],
    "dict": {
      "k1": "v1"
    }
  },
  "k2": 3.1622776601683795,
  "k3": "{\\"a\\": 1, \\"b\\": [1, \\"d\\"]}",
  "k4": "2021-04-27 10:49:54",
  "k5": "helloworld",
  "k6": [
    "12",
    3,
    [
      1,
      2
    ],
    4.5
  ],
  "k8": "application/json",
  "k7": None,
  "k9": 1619491794.670995
}
```

⚠ 注意:

关于 DataWay 中使用的第三方模块函数，可以参见 [函数参考](#)。

HTTP 表单格式

HTTP 表单格式的数据代表 mimeType 为 application/x-www-form-urlencoded 的 Entity 中数据序列化后得到的类型。

- 使用 Entity.from_bytes 方法，则 Dataway 对输入的 str/bytes 类型最终解析成一个 dict 类型数据结构。
- 使用 Entity.from_value 构造方法，支持 dict/MultiMap 两种输入类型，并最终解析成一个 MultiMap 类型数据结构。

对 HTTP 表单格式数据，由于其内部实现为 MultiMap 数据类型，因此除了通用的 Entity 选择器语法，还支持特殊的选择器语法。

Selector	含义
['*key']	返回字典下面的 key 对应的所有元素
['key']	返回字典下面的 key 对应的第一个元素

以下将通过示例来说明如何使用 HTTP 表单格式数据：

构造并使用 HTTP 表单格式的 Entity

本示例将展示如何使用 Entity 的构造函数构造一个 HTTP 表单格式的 Entity，并使用 Entity 选择器语法获取 Entity 的属性和数据。

输入

HTTP 表单格式的输入 `k1=123&k2=helloworld&k3=2&k3=abc` 将被当做 Dataway 输入参数 msg 的 payload 属性中，实际的输入类型将会转换为 MultiMap 数据类型。

```
{
```

```
"k1": 123,
"k2": "helloworld",
"k3": [2, "abc"]
}
```

DataWay 脚本

以下 DataWay 脚本使用 `from_value` 函数将 dict 数据类型转换成 Entity 对象，然后用选择器获取 Entity 的元数据和元素。

```
def dw_process(msg):
    entity = Entity.from_value(msg.payload, mime_type='application/x-www-form-urlencoded', encoding='utf-8')
    return {
        'blob': entity['^blob'],
        'mimeType': entity['^mime_type'],
        'k1': entity['k1'],
        'k3': entity['^value']['k3'],
        'k3multi_selector': entity['^value']['*k3'],
        'k5': entity.get('k5', 'default_value')
    }
```

输出

Dataway 的脚本输出为一个 dict，结果如下：

```
{
  "blob": b'k1=123&k2=helloworld&k3=2&k3=abc',
  "mimeType": "application/x-www-form-urlencoded",
  "k1": 123,
  "k3": 2,
  "k3multi_selector": [2, "abc"],
  "k5": "default_value"
}
```

文本格式

文本格式的数据代表 mimeType 为 text/plain 的 Entity 中数据序列化后得到的类型。无论是 `Entity.from_value` 还是 `Entity.from_bytes` 函数，输入均为 str/bytes 类型，输出为 str 数据类型。

以下将通过示例来说明如何使用文本格式数据：

构造文本格式的 Entity

本示例将展示如何使用 Entity 的构造函数构造一个文本格式的 Entity，并使用 Entity 选择器语法获取 Entity 的属性和数据。

输入

文本格式的输入 "This is a text plain message" 将被当做 Dataway 输入参数 msg 的 payload 属性中，实际的输入类型将会转换为文本数据类型。

DataWay 脚本

以下 DataWay 脚本使用 `from_value` 函数将字符串数据转换成 Entity 对象，然后用选择器获取 Entity 的元数据和元素。

```
def dw_process(msg):
    entity = Entity.from_value(msg.payload, mime_type='text/plain', encoding='utf-8')
    return entity['^value']
```

输出

DataWay 的脚本输出为一个 str 字符串，结果为 "This is a text plain message"。

XML 格式

XML 格式的数据代表 mimeType 为 application/xml 或 text/xml 的 Entity 中数据序列化后得到的类型。

- 使用 `Entity.from_bytes` 方法，则 Dataway 对输入的 str/bytes 类型最终解析成一个 dict 数据类型。

- 使用 Entity.from_value 构造方法，仅支持 dict 输入类型，并最终解析成一个 dict 数据结构。同时，输入的 dict 仅包含一个默认的 key "root"，value 则为内置的 MultiMap，在 MultiMap 中可以自由操作 msg 属性。

以下将通过两个示例来说明如何使用 XML 格式数据：

对 XML 格式数据，除了通用的 Entity 选择器语法，还支持特殊的选择器语法。

Selector	含义
['*key']	返回 xml 某个节点 key 对应的所有元素
['key']	返回 xml 某个节点 key 对应的第一个元素
['#text']	获取 xml 某个节点的文本值
['@attr']	获取 xml 某个节点的 attr 属性值

构造 XML 格式的 Entity

本示例将展示如何使用 Entity 的构造函数构造一个 XML 数据格式的 Entity，并使用 Entity 选择器语法获取 Entity 的属性和数据。

输入

DataWay 输入参数 msg 的 payload 值为常量1，同时 msg.vars 中包含一个 key 为 "abc"，value 为 "123" 的键值对。

```
{
  "payload": 1,
  "vars": {
    "abc": "123"
  }
}
```

DataWay 脚本

以下 DataWay 脚本使用 from_value 函数将 dict 类型转换成 Entity 对象，并返回。

```
def dw_process(msg):
    a = math.floor(1.4)
    return Entity.from_value({
        'root': {
            'k1': msg.vars['abc'],
            'k2': json.dumps('哈哈', ensure_ascii=False),
            'k3': a + 1,
            '@id': "hello",
            '#text': "<a>dwad</a>",
            'k4': ['abc', 'def', None],
        }
    }, mime_type = 'application/xml')
```

输出

DataWay 脚本的输出结果为一个 Entity 类型对象，其中 blob 为符合 XML 语法的二进制数据，如下所示：

```
{
  "mime_type": "application/xml",
  "encoding": "utf-8",
  "blob": b'<?xml version="1.0" encoding="utf-8"?>
<root id="hello"><k1>123</k1><k2>"哈哈"</k2><k3>2</k3><k4>abc</k4><k4>def</k4><k4></k4><a>dwad</a></root>'
}
```

使用 XML 特定选择器

本示例将示范如何在 XML 格式数据中使用特定语法的选择器。

输入

DataWay 输入参数 msg 的 payload 值为常量1，同时 msg.vars 中包含一个key为 "abc"，value 为 "123" 的键值对。

```
{
  "payload": 1,
  "vars": {
    "abc": "123"
  }
}
```

DataWay 脚本

以下 DataWay 脚本使用 from_value 函数将 dict 类型转换成 Entity 对象，然后用选择器获取 Entity 的数据。

```
def dw_process(msg):
    a = math.floor(1.4)
    entity = Entity.from_value({
        'root': {
            'k1': msg.vars['abc'],
            'k2': json.dumps('哈哈', ensure_ascii=False),
            'k3': a + 1,
            '@id': "hello",
            '#text': "<a>dwad</a>",
            'k4': ['abc', 'def', None],
        }
    }, mime_type = 'application/xml')
    return {
        'k1': entity['root']['#text'] + entity['root']['@id'],
        'k2': entity['root']['k1'],
        'k3': entity['root']['*k4']['^value'],
        'k4': entity['root']['k4'],
        'k5': entity['^mime_type']
    }
```

输出

DataWay 脚本的输出结果为一个 dict 类型数据，如下所示：

```
{
  "k1": "<a>dwad</a>hello",
  "k2": "123",
  "k3": ['abc', 'def', None],
  "k4": "abc",
  "k5": "application/xml"
}
```

⚠ 注意：

在 XML 格式数据中，root 节点为默认节点，其属性使用 @id=123 的方式指定，文本使用 #text 的方式指定。value为一个 MultiMap 类型，其中 key 为每一个子节点名称，value 为不同的值。

CSV 格式

CSV 格式的数据代表 mimeType 为 application/csv 的 Entity 中数据序列化后得到的类型。

- 使用 Entity.from_bytes 方法，则 Dataway 对输入的 str/bytes 类型最终解析成一个 list 类型数据结构，其中每一项元素均为一个 dict。
- 使用 Entity.from_value 构造方法，支持 list 输入类型，并最终解析成一个 list 类型数据结构，其中每一项元素均为一个 dict。

以下将通过示例来说明如何使用 CSV 格式数据：

构造 CSV 格式的 Entity

本示例将展示如何使用 Entity 的构造函数构造一个 CSV 格式的 Entity，并使用 Entity 选择器语法获取 Entity 的属性和数据。

输入

DataWay 输入参数 msg 的 payload 值为常量1，同时 msg.vars 中包含一个key为 "abc"，value 为 "123" 的键值对。

```
{
  "payload": 1,
  "vars": {
    "abc": "123"
  }
}
```

DataWay 脚本

以下 DataWay 脚本使用 from_value 函数将 dict 类型转换成 Entity 对象，并使用选择器语法获取 Entity 对象的属性值。

```
def dw_process(msg):
    entity = Entity.from_value([
        {'k1': 'abcd', 'k2': 123.0, 'k3': True},
        {'k1': 'defs', 'k2': 'dwdw,2e', 'k3': 10},
    ], mime_type = 'application/csv')
    return {
        'var1': entity['^blob'],
        'var2': entity['^mime_type'],
        'var3': entity['^encoding'],
        'var4': entity[0]['k2'] + entity[1]['k3'],
        'var5': entity[1]['k2']
    }
```

输出

DataWay 脚本的输出结果为一个 dict 类型数据，如下所示：

```
{
  "var1": "b'k1,k2,k3\r\nabcd,123.0,True\r\ndefs,dwdw,2e',10\r\n'",
  "var2": "application/csv",
  "var3": "utf-8",
  "var4": "133",
  "var5": "dwdw,2e"
}
```

⚠ 注意：

对于 CSV 数据格式，接收的 list 每一项元素均为 dict 类型。每一项元素中的 keys 需保持相同，作为 CSV 文本的标题行；每一项元素中的 values 则代表该行的数据值，用逗号分隔。

HTTP Form-Data 表单

HTTP Form-Data 表格式的數據代表 mime-type 为 multipart/form-data 的 Entity 中数据序列化后得到的类型。

浏览器中的 multipart/form-data

在浏览器发送 Content-Type 为 multipart/form-data 的请求时，实际传输的 byte 数组转换成字符串如下所示：

- 每一项参数由一个 boundary 开头，标识着这一项的开始，例如：--34b21。

⚠ 注意：

-- 为固定开头，34b21 为浏览器随机的一个不超过70位的字符串。

- boundary 开头下是 Content-Disposition 和 Content-Type 的固定 headers，其中：
 - Content-Disposition 包含 name 和 filename 两项，name 为参数名，filename 为文件名。
 - Content-Type 即为输入内容的 Content-Type。

- 如果 filename 为 "*.txt" 或者为空，则 Content-Type 默认为 text/plain；如果 filename 为其他，则 Content-Type 会根据后缀名自动判断。同时，也支持其他的扩展 headers。
- 再往下是实际的内容，如果 Content-Type 为 text/plain，则为普通字符串，例如："Book"；如果 Content-Type 为 application/json，则为一个 json 类型的字符串，如下方的 file1 对应的 json 结构。
- 最后用 --34b21-- 来标识该段请求的结束。

```
--34b21
Content-Disposition: form-data; name="text"
Content-Type: text/plain

Book
--34b21
Content-Disposition: form-data; name="file1"; filename="a.json"
Content-Type: application/json

{
  "title": "Java 8 in Action",
  "author": "Mario Fusco",
  "year": 2014
}
--34b21
Content-Disposition: form-data; name="file2"; filename="a.html"
Content-Type: text/html

<!DOCTYPE html>
<title>
Available for download!
</title>
--34b21--
```

Form-Data 的构造和使用

- 使用 Entity.from_bytes 方法，则 Dataway 对输入的 str/bytes 类型最终解析成一个 FormDataParts 数据结构。
- 使用 Entity.from_value 构造方法，支持 Entity、Formdata、list/dict 输入类型，并最终解析成一个 FormDataParts 数据结构。

⚠ 注意：

- 如果为 Entity 类型，则会首先判断 Entity['mime_type'] 是否为 multipart/form-data。如果为是，直接返回该 Entity；否则将报错。
- 如果为 FormdataParts 类型（即上文提到的 map+ 数组结构），则直接赋值给 Entity。
- 如果为 list/tuple 类型，则需要保证为以下数据结构：每一项的第一个元素为参数名。第二个元素如果为 list/tuple，则第二个元素的第一项为文件名，第二项为实际的内容，第三项为 Content-Type，第四项为一个 dict，代表 extra_headers；如果第二个元素类型为 str，则文件名为空，实际内容即为字符串内容，Content-Type 默认为 text/plain。

对 HTTP Form-Data 格式数据，除了通用的 Entity 选择器语法，还支持特殊的选择器语法。

Selector	含义
['parts']	返回自定义类型 FormDataParts 类型
['parts'][0]	返回 FormData 的第0项
['parts']['a']['content']	返回 FormDataParts 中 key 为'a'的 value 中的 content 值
['boundary']	返回 FormDataParts 的分隔符

应用示例

以下将通过两个示例来说明如何使用 HTTP FORM-DATA 表单格式数据。

示例一：构造 HTTP Form-Data 格式的 Entity

输入

以下 Dataway 脚本使用 `from_value` 函数将 dict 类型转换成 Entity 对象，并返回。

Dataway 脚本的输出结果为一个 Entity 类型对象，其中 blob 为符合 multipart/form-data 结构的二进制数据，如下所示。

本示例将示范如何在 FORM-DATA 格式数据中使用特定语法的选择器。

输入

Dataway 输入参数 msg 的 payload 为一个 Entity 类型，其 mime_type 为 multipart/form-data，通过 Entity.from_value 函数创建。

```
def dw_process(msg):
    return Entity.from_value(
        [
            ('a', ('test', '1233', 'application/json', {'test111':1})),
            ('b', (None, '2333', 'text/plain')),
        ],
        mime_type="multipart/form-data; boundary=123"
    )
```

Dataway 脚本

以下 Dataway 脚本将使用 FORM-DATA 选择器对 msg.payload 进行处理，并返回一个 dict。

```
def dw_process(msg):
    return {
        'k1': str(type(msg.payload)),
        'k2': msg.payload['^mime_type'],
        'k3': msg.payload['^encoding'],
        'k4': msg.payload['^raw'],
        'k5': msg.payload['boundary'],
        'k6': msg.payload['parts']['a']['headers']['Content-Disposition']['name']
            + '-' + msg.payload['parts'][1]['headers']['Content-Type']
    }
```

输出

Dataway 脚本的输出结果为一个 dict 类型数据，结果如下。

```
{
  'k1': 'Entity',
  'k2': 'multipart/form-data;boundary=123',
  'k3': 'utf-8',
  'k4': b'--123
Content-Disposition: form-data; name="a"; filename="test"
Content-Type: application/json
test111: 1

1233
--123
Content-Disposition: form-data; name="b"
Content-Type: text/plain

2333
--123--
',
  'k5': '123',
  'k6': 'a-text/plain'
}
```

其他类型

对其他类型的 mime_type，Dataway 不支持直接用 Entity.from_value 函数构造，但支持从上游读取数据，以及使用 Entity.from_bytes 函数构造一个封装的 Entity。

下面将通过一个示例来说明，假设输入数据为一个二进制 byte 流，我们通过 Set Payload 组件中使用 Dataway 表达式将该二进制 byte 流封装到 msg.payload 中，然后在下游可以使用 [Entity 选择器](#) 语法进行操作。

Dataway 表达式：

```
def dw_process(msg):
    b = msg.payload
    return Entity.from_bytes(b, mime_type='application/octet-stream')
```

DataWay 脚本调试

最近更新时间：2023-06-08 14:48:05

在开发 Dataway 时，对脚本进行调试和测试，以方便问题排查和结果验证，腾讯云数据连接器平台提供 Dataway 脚本的调试功能。该功能可通过手工定义输入参数 msg，单击测试后可以直接查看脚本运行结果、调试日志和错误信息。

操作步骤：

1. 打开腾讯云数据连接器平台，新建一个组件，在 Dataway IDE 编辑框中输入表达式。

① 说明

Dataway 调试模式下，支持在表达式中通过 print() 函数打印要观察的信息，运行结束后打印消息会显示在界面上。



2. 单击 Dataway 表达式编辑框右上角“Debug”，弹出 msg 数据填写对话框，在对话框可以对 message 的 payload、vars 和 attrs 进行设置。设置完成后单击开始测试，系统会自动组装成一个 msg 参数，并作为脚本的输入传递到 dw_process 函数中。

请模拟表达式的输入数据

×

payload 值

entity.json

{ "var1": "helloworld" }

attribute 值

变量名	变量值	操作
+添加变量		

variable 值

变量名	变量值	操作
b	string 123	删除
c	string 456	删除
+添加变量		

开始测试

取消

3. 运行完成 dw_process 函数后，编辑框下方会弹出运行结果和 print 调试日志，如果运行错误会有 error 报错信息。

- 输入代表 Dataway 表达式的运行结果，日志代表在脚本中使用 print 函数打印的调试日志，错误代表脚本运行错误，运行正确无错误显示绿色对勾标志。

Dataway表达式 ⓘ

1

def dw_process(msg):

2

print(msg.vars)

3

print(l)

4

return msg.payload['var1'] + msg.vars['b']

输出 ✓

错误

日志

"helloworld123"

自动联想

payload

payload

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["statusCode"]

attributes

attrs["reasonPhrase"]

attributes

attrs

vars

vars["a"]

vars

vars["b"]

确定

取消

Dataway表达式 ②

```
1 def dw_process(msg):
2   print(msg.vars)
3   print(1)
4   return msg.payload['var1'] + msg.vars['b']
```

输出  错误 日志

```
{'b': '123', 'c': '456'}
1
```

确定

取消

自动联想

payload

payload

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["statusCode"]

attributes

attrs["reasonPhrase"]

attributes

attrs

vars

vars["a"]

vars

vars["b"]

4. 通过添加 print 日志和模拟请求 msg 参数的数据，可完成对 Dataway 表达式的调试和验证工作。

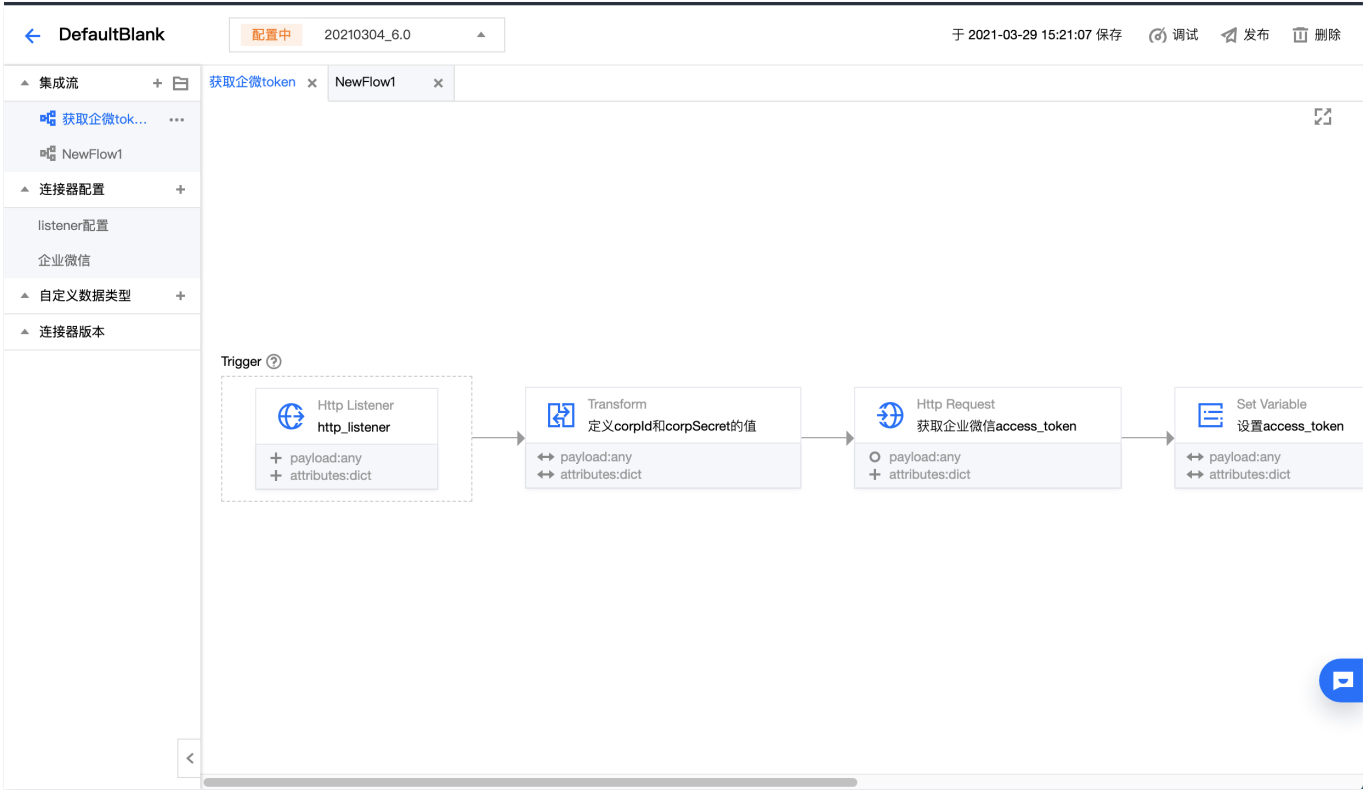
场景案例

最近更新时间：2024-10-18 16:32:52

本节将通过几个典型的场景案例说明 Dataway 在腾讯云数据连接器平台中应用。

案例一：使用 Dataway 获取 token

腾讯云数据连接器中封装企业微信连接器，可以完成成员管理、消息推送等多种操作，每次操作本质上都是调用 [企业微信 API](#) 完成的。在调用 API 前，需要获取 access_token。相当于创建一个登录凭证，其他的业务接口都依赖 access_token 鉴权调用者身份。
本示例将模拟该调用过程，并说明 Dataway 脚本在其中起到的作用。



- 上图为整个流的配置图，其中 Transform 组件使用 Dataway 表达式设置 corpId 和 corpSecret 两个 variable 参数。

Transform ?

配置

备注

输入信息

输出信息 corpId x corpse... x +

payload any

attributes dict

- method string
- host string
- requestUri string
- requestPath string
- queryParams multimap
- uriParams multimap
- headers multimap

cookies dict

- remoteAddr string

variables dict

Dataway表达式 ?

1 def dw_process(msg):

2 return 'test-corpId'

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

图形转换

脚本输入

Transform ?

配置备注

输入信息

输出信息

corpId x corpse... x +

payload any

attributes dict

method string

host string

requestUri string

requestPath string

queryParams multimap

uriParams multimap

headers multimap

cookies dict

remoteAddr string

variables dict

Dataway表达式 ?

1 def dw_process(msg):

2 return 'test-corpsecreat'

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

图形转换

脚本输入

- HTTP Request 组件向企业微信 API 地址发起 get-token 请求调用，请求参数为 msg.vars 中的 corpId 和 corpSecret。在 HTTP Request 组件的请求参数中，使用 Dataway 表达式进行动态设置。

HTTP Request ?

...

☐

当前使用的是 企业微信 物理配置

编辑 | 切换

配置

备注

基本配置

请求路径 *

string ▼

/cgi-bin/gettoken

f(x)

指定发送请求的路径

请求方法 *

POST ▼

f(x)

指定发送请求的方法

请求头

def dw_process(msg): return

🗑

指定发送请求的请求头,支持表达式

请求参数

def dw_process(msg): params = {'corpid': msg

🗑

指定发送请求的请求参数, 只支持表达式, 返回字典类型

请求体

string ▼

请输入字符串

f(x)

指定发送请求的请求体, 支持字面量和表达式

高级配置

输出消息绑定自定义数据类型

消息属性

payload ▼

类型选择

空 ▼

+

Dataway表达式

```

1  def dw_process(msg):
2      params = {'corpid': msg.vars['corpid'],
3                'corpsecret': msg.vars['corpsecret']}
4      }
5      return params

```

自动联想

payload

payload

attributes

attrs["method"]

attributes

attrs["host"]

attributes

attrs["requestUri"]

attributes

attrs["requestPath"]

attributes

attrs["queryParams"]

attributes

attrs["uriParams"]

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["remoteAddr"]

attributes

attrs

vars

确定

取消

- Set-Variable 组件将获取到 HTTP response 解析，获取 access_token 后存到 msg.vars 中，使用 Dataway 动态赋值。

Set Variable ?

配置

备注

配置 / 编辑

Dataway表达式 ?

1

def dw_process(msg):

2

return msg.payload.get(

3

'access_token', '')

自动联想

payload

payload

attributes

attrs["headers"]

attributes

attrs["cookies"]

attributes

attrs["statusCode"]

attributes

attrs["reasonPhrase"]

attributes

attrs

vars

vars["corpid"]

vars

vars["corpsecret"]

确定

取消

- Cache 组件通过键值对保存 access_token，以方便下次无需请求直接从缓存中获取。Cache 的数据使用 Dataway 表达式获取。

 cache  存储

...



配置

备注

通用

▲

键值 *

string ▼

cache_key_access_token_qw

f(x)

唯一标识，可用来读取数据

数据 *

string ▼

def dw_process(msg): return ms



需要存储的数据

存储时长(秒) *

string ▼

3600

f(x)

存储时长，取值范围为1~2592000

高级配置

▲

输出消息绑定自定义数据类型

消息属性

payload

▼

类型选择

空

▼

+

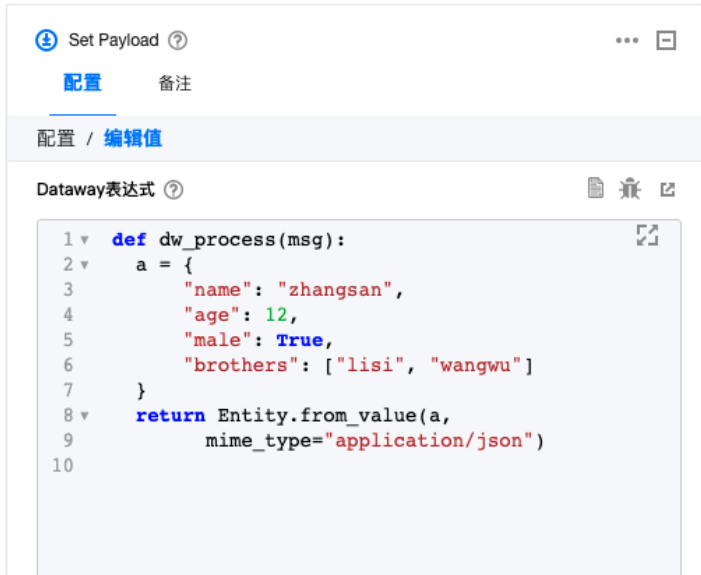
实际上企业微信连接器使用 XML 语言完成 get-token，本示例为了方便查看在流中复现该操作。Dataway 表达式在其中主要的作用是通过操作流转消息 Message，实现动态赋值，并交给上层调用组件使用。

案例二：将 JSON 类型输入转换成 XML 类型输入

使用 Dataway 表达式可以完成数据类型的转换，例如：JSON 转换成 XML、XML 转换成 CSV 等。本示例将通过一个将 JSON 类型的输入转换成 XML 的示例，说明 Dataway 的数据转换作用。

输入

我们在流中预先使用 Set Payload 组件对 Message 的 payload 属性设置成封装了 JSON 格式的 Entity 数据。



Dataway 表达式

在 Set Payload 组件后新增 Transform 组件，使用 Dataway 表达式重新设置 Message 消息的 payload。

```
def dw_process(msg):
    input_param = msg.payload['^value']
    return Entity.from_value({
        'root': {
            'name': input_param['name'],
            'age': input_param['age'],
            'male': input_param['male'],
            'brothers': input_param['brothers'],
            "@id": "haha"
        }
    }, mime_type="application/xml")
```

输出

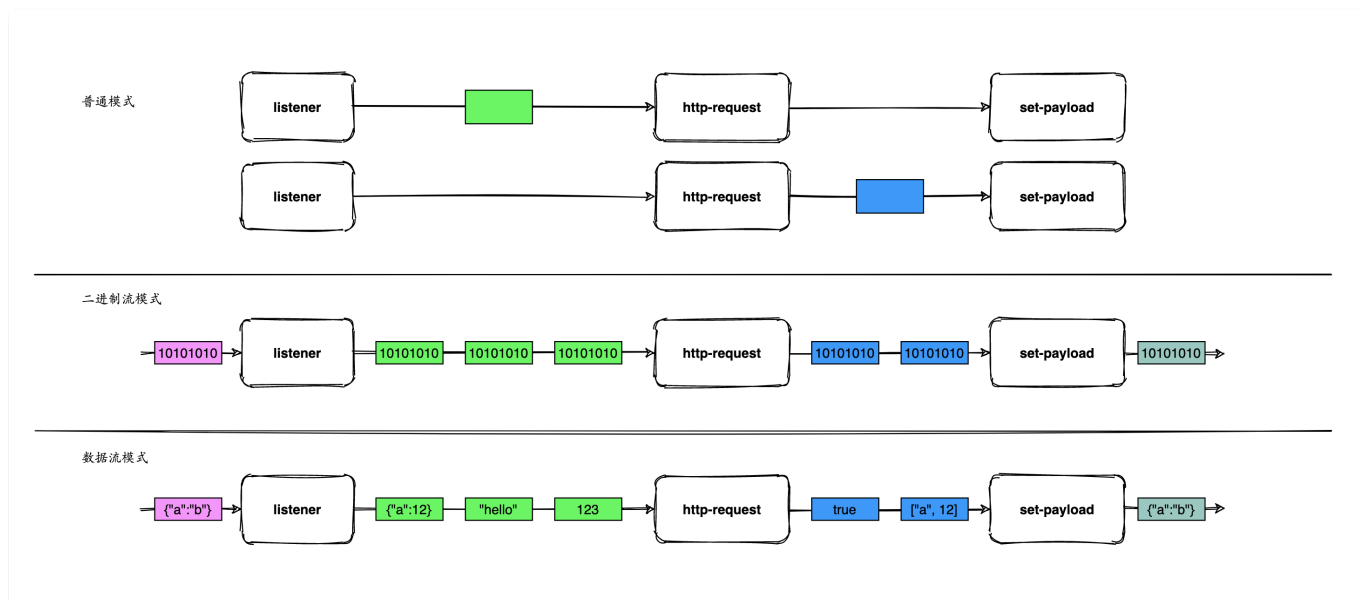
Transform 组件将 Message 的 payload 设置为 Entity 类型，使用 Entity['^blob'] 获取二进制数据并解码，可以查看输出的 XML 结构字符串，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<root id="haha">
  <name>zhangsan</name>
  <age>12</age>
  <male>true</male>
  <brothers>lisi</brothers>
  <brothers>wangwu</brothers>
</root>
```

Dataway 表达式通过 Entity 类型提供强大的数据转换功能，通过设置不同的 mime_type 参数，可以实现不同类型的数据转换。更多详细内容请参见 [使用 Entity 对象](#)。

案例三：Dataway 处理流式数据

流式数据指数据从流的开始到结束，过程是流式的，数据不需要全读入内存。这样的好处是能支持大文件且可以做到实时处理，因此在腾讯云数据连接器中常用于实时、格式化的处理大数据。



本示例将说明如何读取流式的输入和产生流式的输出。

产生流式输出

使用 Dataway 表达式可以产生流式的输出。在 Set Payload 组件中使用如下 Dataway 表达式：

Dataway 表达式

```
def dw_process(msg):
    for x in range(10):
        yield x
```

输出

Dataway 表达式将产生一个流式对象，并设置到 payload 中，供下游使用。

读取流式输入

使用 Dataway 表达式可以读取上一步产生的流式输入。msg.payload 为一个流式对象，可以使用 for 循环读取该流式对象，并进行数据处理。

Dataway 表达式

```
def dw_process(msg):
    sum = 0
    for x in msg.payload:
        sum += x
    return sum
```

输出

Dataway 表达式使用流式处理输入数据，并最终返回 45。

错误码说明

最近更新时间：2023-06-08 14:48:05

DataWay 脚本运行错误时，会打印出错误日志。如下图所示，主要结构为 ErrorCode 和 ErrorMsg 两部分。其中 ErrorCode 为错误码，ErrorMsg 为错误信息。



错误码说明

错误码	错误信息
1000000	表达式最大长度超限
1000001	表达式语法校验失败
1200000	表达式执行失败
1200001	表达式执行失败
1200002	表达式语法校验失败
其他	内部错误

运行时异常说明

错误码为 1200000 和 1200001 的错误码进行说明，代表表达式在执行过程中的脚本错误。其主要结构为：TraceBack + 异常说明。TraceBack 为 Dataway 表达式的异常栈，Type 为一种异常类型。常见的异常类型如下表所示：

异常类型	说明
SyntaxError	Dataway 脚本语法错误，例如拼写、缩进错误等
IndexError	下标索引超出序列边界，例如：当 x 只有三个元素，却试图访问 x[5]
KeyError	试图访问字典里不存在的键
NameError	使用一个未定义的变量
TypeError	传入类型与要求的不符
ValueError	传入一个不期望的值
AttributeError	试图访问一个对象没有的属性，例如：foo.x，但是 foo 没有 x 属性

常见问题

最近更新时间：2024-10-18 16:23:31

DataWay 表达式与 Python 脚本是什么关系？

DataWay 表达式基于 Python3 语言进行封装，在 Python 的基础上进行功能的裁剪，以及要求必须定义合法的入口函数 `dw_process (msg)`。因此，所有的 DataWay 表达式正常执行结束之后都会有最终的返回值，但 Python 脚本 可以没有明确返回值。

DataWay 表达式在什么场景下使用？

目前几乎所有的腾讯云数据连接器核心组件都具有表达式求值的能力，DataWay 是腾讯云数据连接器具有动态求值能力的关键所在。

DataWay 脚本运行错误怎么解决？

1. 首先排除 DataWay 脚本的语法错误，确保在 DataWay 编辑时通过语法检查。
2. 通过查看运行错误日志，定位 DataWay 脚本的错误原因及脚本错误行数，进行问题分析和定位。同时，也可通过 [DataWay 脚本调试](#) 进行问题排查。

如何使用其他第三方模块？

DataWay 默认支持的内置模块仅包含 `time`、`json`、`math`、`base64`、`hmac`、`random`、`hashlib`、`Crypto`、`socket`、`struct`、`decimal`、`urllib`、`csv` 和 `datetime` 模块。

一般而言，第三方模块对 DataWay 脚本而言已足够。如果实际情况需要新增第三方模块，请联系我们。经安全评估后，会将该模块开放。

Cron 表达式介绍

最近更新时间：2024-12-11 16:22:02

Quartz Cron 表达式是一种广泛应用于描述时间和日期信息的标准表达式。当在集成流 Scheduler 组件中配置的 Cron 表达式与您的日期时间相匹配时，系统将自动触发集成流的运行。

Cron 表达式结构

Cron 表达式是一个字符串，由6个必要的设置组成，同时支持可选的年份设置。
通常 Cron 表达式设置由空格分开，主要结构为：秒 分 小时 日期 月份 星期（年，可选）。

字段说明

字段	允许值	允许的特殊字符	是否必须
秒（Seconds）	0 – 59的整数	, - * / 四个字符	是
分（Minutes）	0 – 59的整数	, - * / 四个字符	是
小时（Hours）	0 – 23的整数	, - * / 四个字符	是
日期（DayofMonth）	1 – 31的整数（需要考虑当月天数）	, - * ? / 五个字符	是
月份（Month）	1 – 12的整数（JAN、FEB、MAR、APR、MAY、JUN、JUL、AUG、SEP、OCT、NOV、DEC）	, - * / 四个字符	是
星期（DayofWeek）	1 – 7的整数或者（SUN、MON、TUES、WED、THUR、FRI、SAT（1=MON，7=SUN））	, - * ? / 五个字符	是
年（Year）	1970 – 2099的整数	, - * / 四个字符	否

特殊字符说明

- 每一个字段都使用数字，还可出现如下特殊字符，它们的含义是：
- *：表示匹配任意值。假如在分字段使用 *，即表示每分钟都将执行一次。
 - ?：表示无指定字段，只能用在日期和星期两个字段。
 - ：表示匹配范围。例如在分钟字段使用 5-20，表示在5分到20分钟之间，每分钟执行一次。
 - /：表示从起始时间开始执行，每隔固定时间执行一次。例如在分钟字段使用 5/20，表示从5分开始，每隔20分钟执行一次，也即25分、45分各执行一次。
 - ,：表示列出枚举值。例如：在分钟字段使用 5,20，表示在5分和20分各执行一次。

示例

表达式	解释
0 0 12 * * ?	每天中午12:00执行一次
0 15 10 ? * *	每天上午10:15执行一次，其中 0 15 10 ? * *、0 15 10 * * ?、0 15 10 * * ? * 三个表达式效果相同
0 15 10 * * ? 2020	在2020年的每天的10:15执行一次
0 * 14 * * ?	每天的14:00 – 14:59时间段，每隔1分钟执行一次
0 0/5 14 * * ?	每天的14:00 – 14:55时间段，每隔5分钟执行一次
0 0/5 14,18 * * ?	每天的14:00 – 14:55，18:00 – 18:55，每隔5分钟执行一次
0 0-5 14 * * ?	每天的14:00 – 14:05时间段，每隔1分钟执行一次
0 10,44 14 ? 3 WED	三月份每周三的下午14:10和14:44各执行一次

0 15 10 ? * MON-FRI	每周周一至周五每天上午10:15执行一次
0 15 10 15 * ?	每个月的第15天的上午10:15执行一次
0 0 12 1/5 * ?	每月1号开始，每隔5天每天中午12:00执行一次
0 11 11 11 11 ?	每年的11月11日11:11执行一次

访问管理示例

最近更新时间：2023-06-08 14:48:05

使用场景

腾讯云数据连接器（APP Flow）是腾讯云控制台内的一款产品，使用腾讯云账号登录，支持 [腾讯云（Cloud Access Management, CAM）](#) 进行访问管理。您可以从腾讯云数据连接器产品界面跳转到腾讯云 CAM 中为您的腾讯云主账号添加子账号，腾讯云数据连接器会自动拉取当前主账号下的所有子账号作为普通成员进入成员列表进行访问。主账号作为系统管理员进入成员列表进行访问。您可以通过使用访问管理（Cloud Access Management, CAM）策略让用户拥有在腾讯云数据连接器（APP Flow）控制台中查看和使用特定资源的权限。本文提供查看和使用特定资源的权限示例，指导用户如何使用控制台的特定部分的策略。

	腾讯云数据连接器权限控制	访问管理（CAM）权限控制	联系	区别
角色	系统管理员（拥有腾讯云数据连接器最高使用权限）	主账户（拥有账户下所有资源的全部权限）	腾讯云数据连接器是腾讯腾讯云控制台内的一款产品，使用腾讯云账号登录，支持 腾讯云（Cloud Access Management, CAM） 进行访问管理。您可以在腾讯云数据连接器产品界面跳转到腾讯云访问管理中为您的腾讯云主账号添加子账号，腾讯云数据连接器会自动拉取当前主账号下的所有子账号作为普通成员进入成员列表，拉取当前主账户作为系统管理员进入成员列表。	访问管理（CAM）和腾讯云数据连接器权限是两套独立的权限控制。访问管理（CAM）是针对腾讯云账号维度的权限控制。腾讯云数据连接器权限只在腾讯云数据连接器的集成项目层面生效。
	项目管理员（拥有集成项目的最高权限）			
	普通成员（基本的读写权限和默认项目权限）	子账户（根据不同授权具备不同权限）		
权限	设置/取消系统管理员	腾讯云数据连接器只读访问权限		
	设置/取消项目管理员	腾讯云数据连接器全读写访问权限		
	设置普通用户的单个应用权限为只读/编辑/编辑并发布	自定义策略		

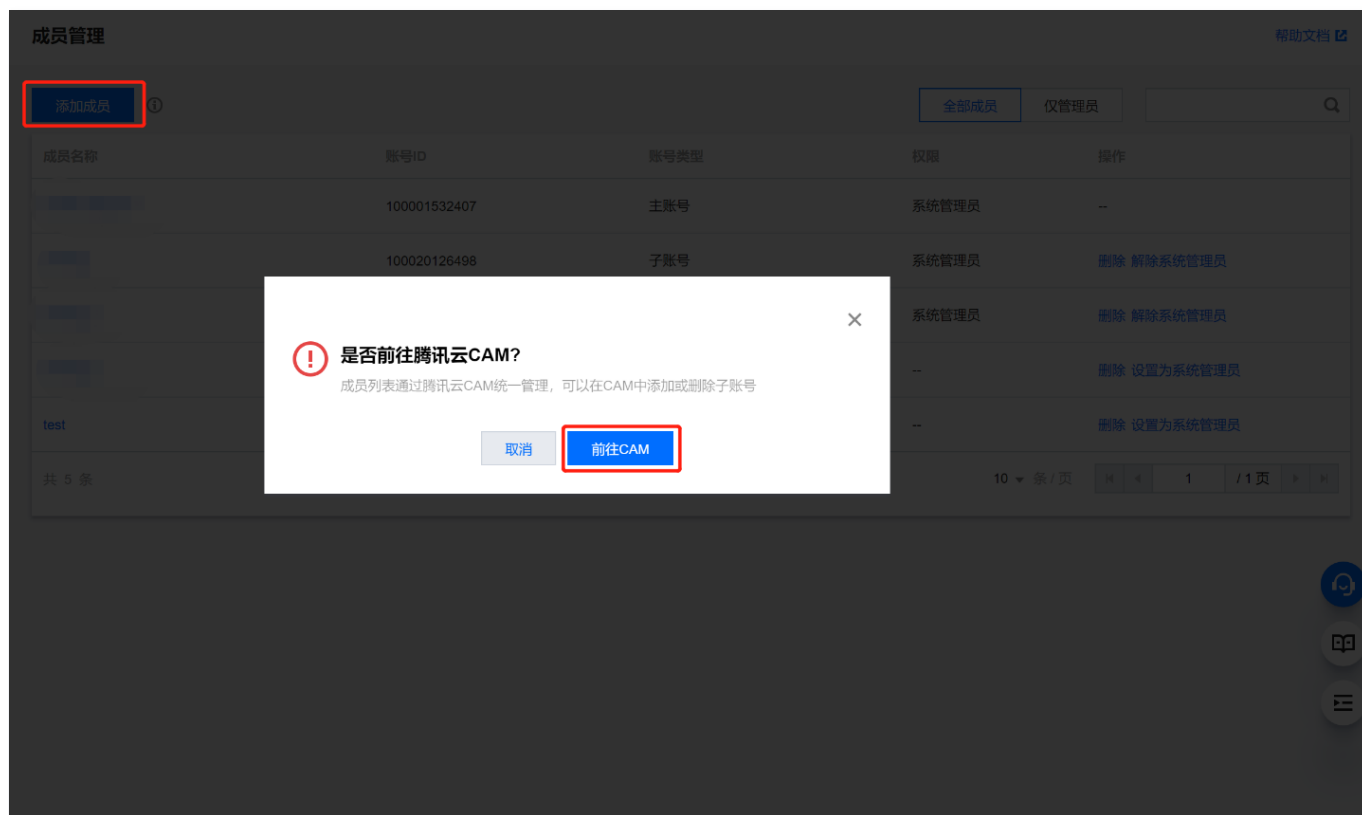
操作示例

添加子用户

引导用户添加子用户并授权腾讯云数据连接器，成为腾讯云数据连接器中的普通成员。操作步骤如下：
登录 [腾讯云数据连接器控制台](#)，选择[成员管理](#) > [添加成员](#)，单击[前往 CAM](#) 跳转至用户列表界面进行操作。

 **注意**

拥有管理员权限（AdministratorAccess）或拥有 CAM 全部权限（QcloudCamFullAccess）的子账号才能创建子用户。创建子用户请参见 [新建子用户](#)。

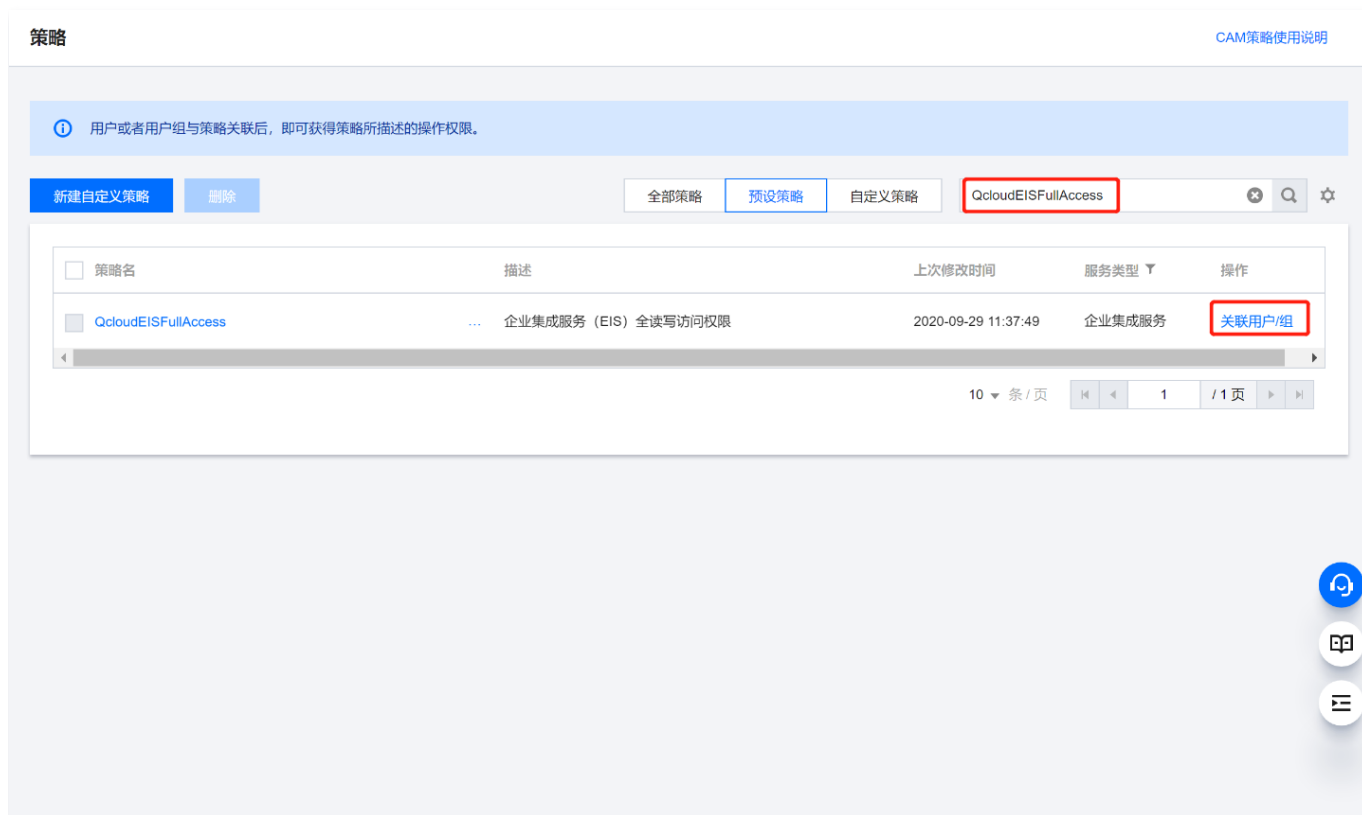


子用户授权管理

子用户授权管理请参见 [授权管理](#)。

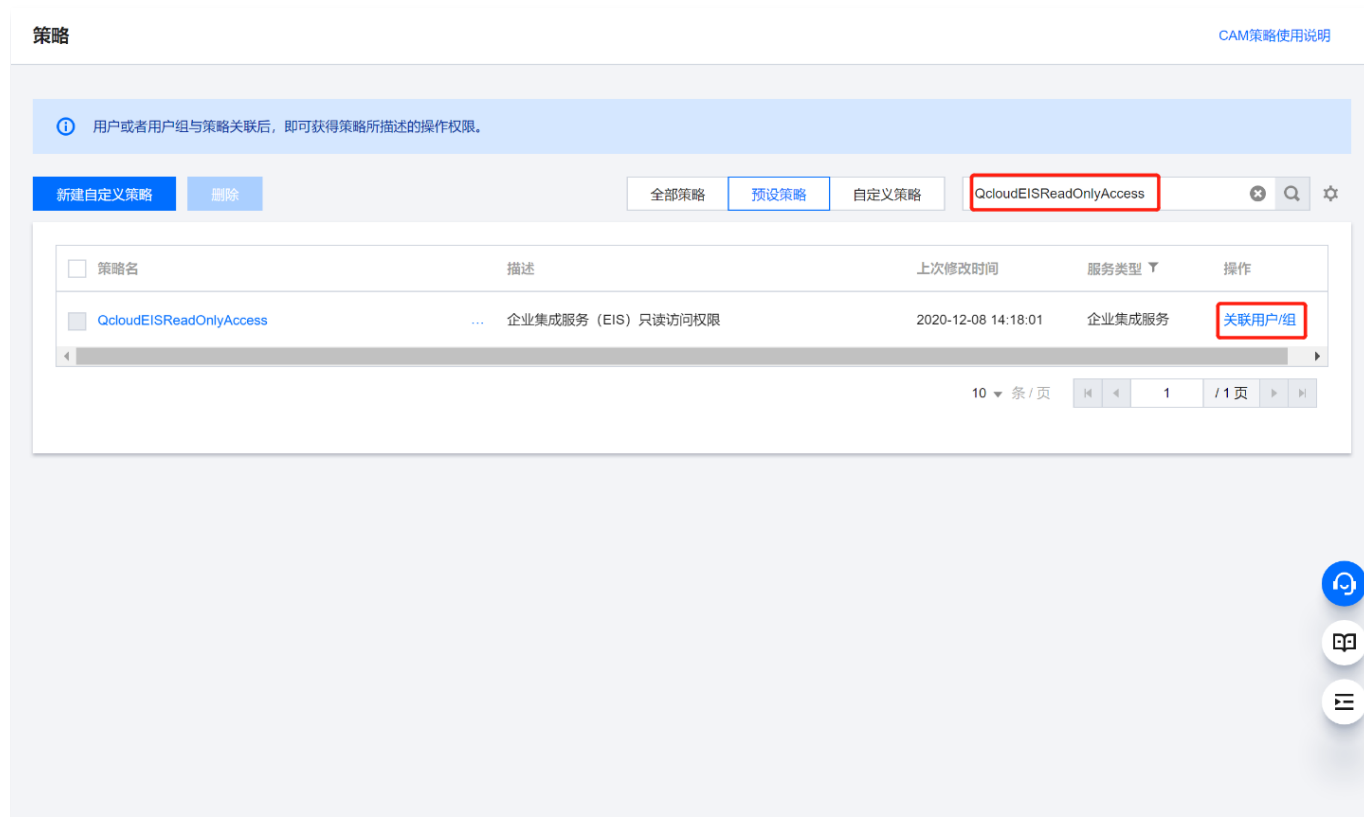
腾讯云数据连接器的全读写策略

如果您希望用户拥有创建和管理腾讯云数据连接器的权限，您可以对该用户使用名称为：QcloudEISFullAccess 的策略。该策略是通过让用户分别对腾讯云数据连接器所有接口和资源都具有操作的权限来达到目的。您可以参见 [授权管理](#)，将预设策略 QcloudEISFullAccess 授权给用户。



腾讯云数据连接器的只读策略

如果您希望用户拥有查询腾讯云数据连接器的权限，但是不具有创建、删除、修改的权限，您可以对该用户使用名称为：QcloudEISReadOnlyAccess 的策略。该策略是通过让用户分别对腾讯云数据连接器中所有以单词“Describe”、“Inquiry”开头的所有操作具有操作的权限来达到目的。您可以参见 [授权管理](#)，将预设策略 QcloudEISReadOnlyAccess 授权给用户。



自定义策略授权

如果您希望用户可以查看并检索腾讯云数据连接器控制台中运行日志的相关操作，可先将以下操作添加到您策略中，再将该策略关联到该用户。

- DescribeHistoryLog 搜索历史日志
- PullLog 拉取实时日志
- SearchLog 搜索历史日志

操作步骤

1. 根据 [策略](#)，创建一个自定义策略。该策略只允许用户在腾讯云数据连接器控制台中查询实时和历史运行日志，其余读写查看列表操作均不允许。
 - 策略内容可参考以下策略语法进行设置（JSON）：

```
{
  "version": "2.0",
  "statement": [
    {
      "effect": "allow",
      "action": [
        "eis:DescribeHistoryLog",
        "eis:PullLog",
        "eis:SearchLog"
      ],
      "resource": [
        "*"
      ]
    }
  ]
}
```

- 策略内容可参考以下策略语法进行设置（可视化策略生成器）：

← 按策略生成器创建

1 编辑策略

>

2 关联用户/用户组

导入策略语法

可视化策略生成器

JSON

▼ 企业集成服务 (3 个操作)

删除

效果 (Effect) *

☒ 允许

☐ 拒绝

服务 (Service) *

企业集成服务 (eis)

操作 (Action) *

读操作 编辑

DescribeHistoryLog
搜索历史日志
列表操作 编辑

PullLog
拉取实时日志

SearchLog
搜索历史日志

资源 (Resource) *

全部资源 (*)

条件 (Condition)

☐ 来源 IP ①

添加其他条件

+ 添加权限

下一步

2. 找到创建的策略，在该策略行的“操作”列中，单击关联用户/组。

策略

CAM策略使用说明

①

用户或者用户组与策略关联后，即可获得策略所描述的操作权限。

新建自定义策略

删除

全部策略

预设策略

自定义策略

搜索策略名称/描述/备注(多关键词空格隔开)

☐ 策略名	描述	上次修改时间	服务类型 ▾	操作
☐ EIS-test	...	2021-09-07 15:27:49	-	删除 关联用户/组
☐ EIS-test	...	2021-08-30 20:07:19	-	删除 关联用户/组

10 ▾ 条 / 页

⏮

⏪

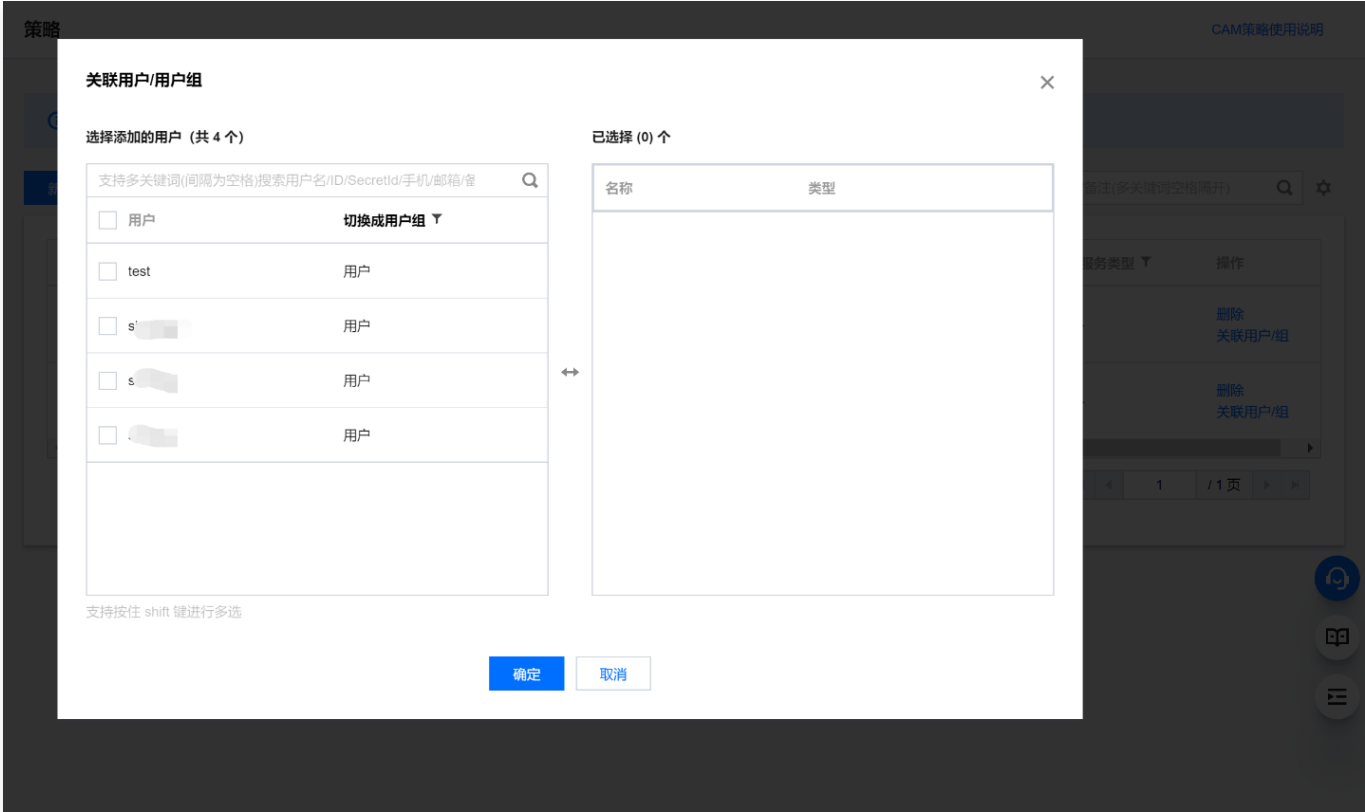
1

⏩

⏭

1 / 1 页

3. 在弹出的“关联用户/用户组”窗口中，选择您需要授权的用户/组，单击确定即可。



权限管理授权

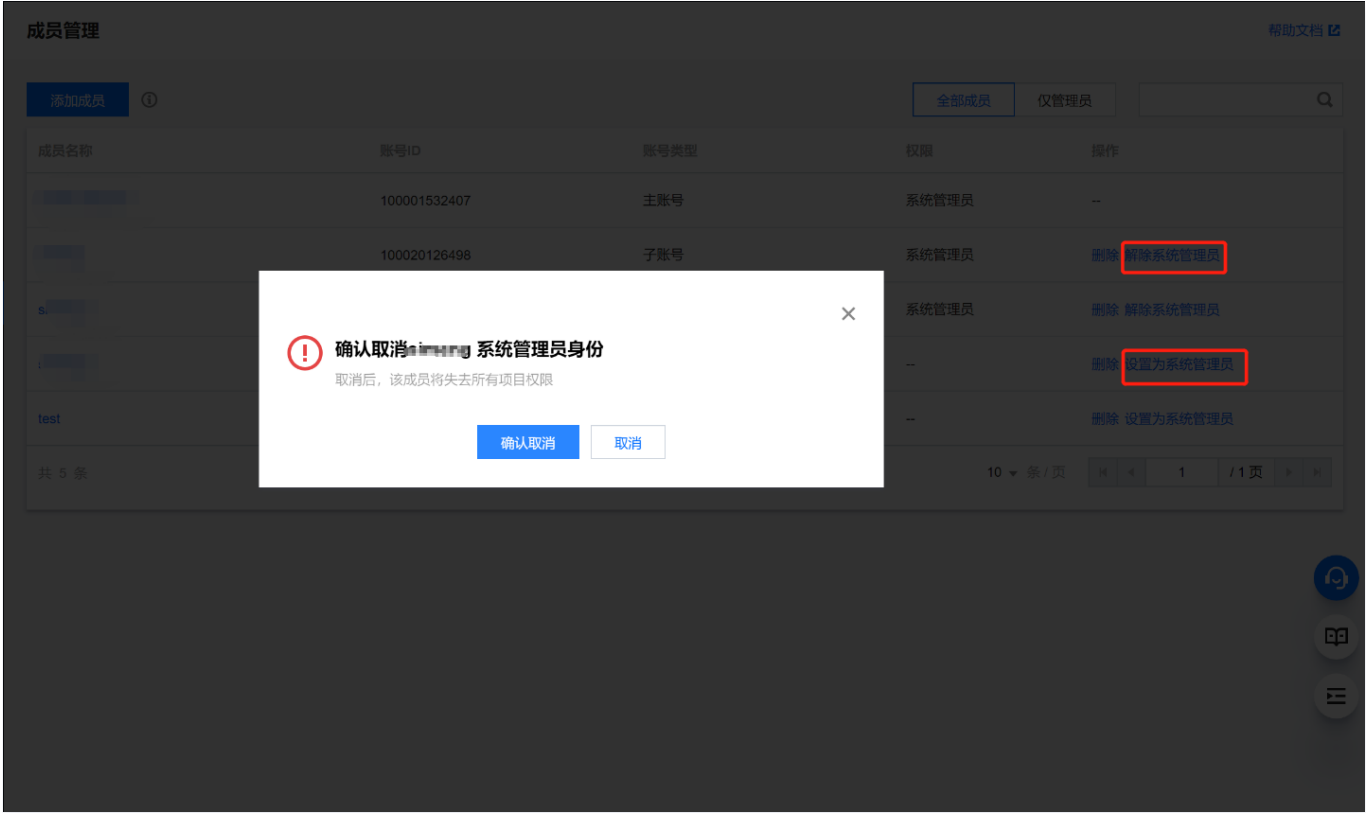
授权用户拥有腾讯云数据连接器项目的操作权限。项目用于腾讯云数据连接器用户区分业务场景和权限划分，用户可以以项目维度管理集成应用。项目支持查看详情、修改名称和删除操作。

项目管理权限

角色	权限
系统管理员	拥有腾讯云数据连接器的最高使用权限，腾讯云主账号自动拥有系统管理员权限且无法撤销。系统管理员可授予、撤销任意成员系统管理员和项目管理员权限，可授予项目成员中任意普通用户对该集成项目中的某些集成应用的只读或读写权限。
项目管理员	拥有子用户腾讯云数据连接器相关权限和集成项目的最高权限，项目管理员可以对该集成项目下所有普通用户授予某些集成应用的只读或读写权限，可以授予或撤销项目管理员权限。
普通成员	拥有子用户腾讯云数据连接器相关权限和默认项目权限，腾讯云主账号下的所有子账号自动为腾讯云数据连接器平台的普通成员。

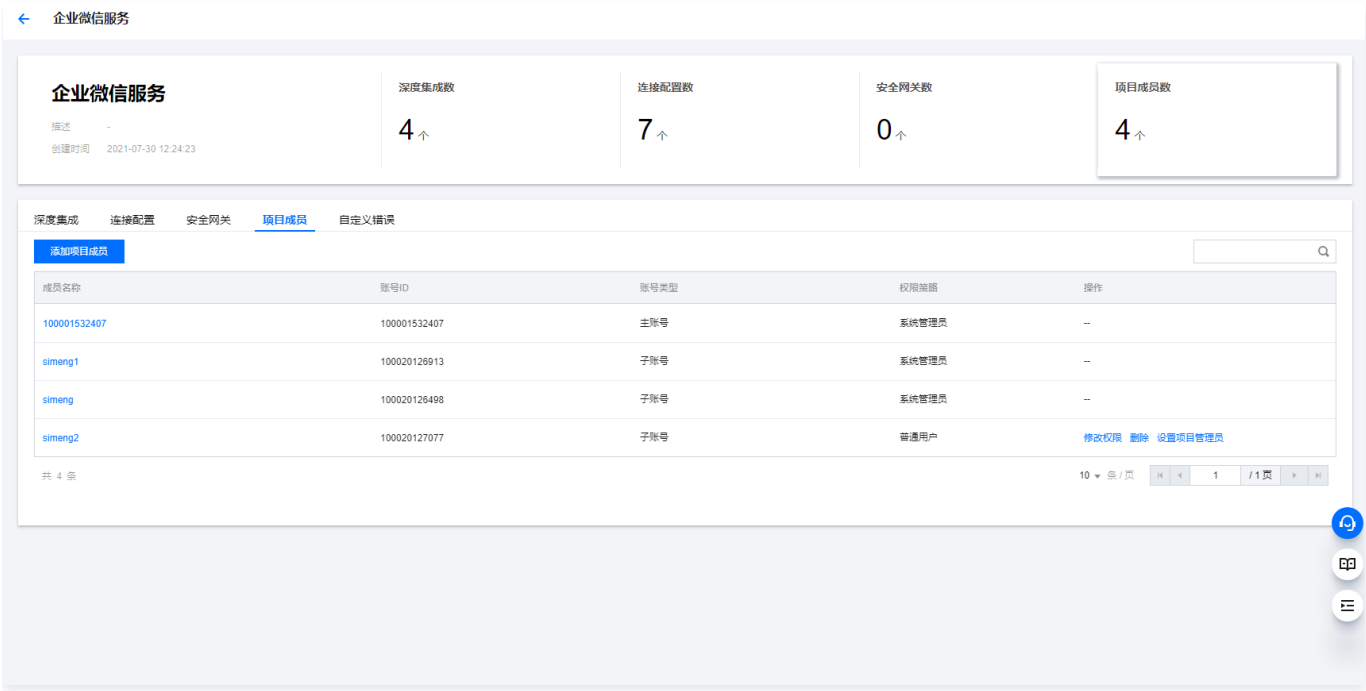
设置/解除系统管理员

系统管理员，可以对其他子账户操作设置/解除系统管理员。该腾讯云主账户默认为系统管理员且不可撤销。登录 [腾讯云数据连接器控制台](#)，选择成员管理 > 设置为/解除系统管理员。



设置/解除项目管理员

系统管理员和项目管理员可以对其他子账户操作设置/解除项目管理员。登录 [腾讯云数据连接器控制台](#)，选择集成项目 > 项目成员管理，可自行配置项目成员相关权限。



授权用户特定操作权限

如果您希望授权用户特定腾讯云数据连接器应用的操作权限，可使用腾讯云数据连接器内部的项目管理功能。系统管理员和项目管理员可以对其他子账户操作修改项目内其他应用的只读或读写权限。登录 [腾讯云数据连接器控制台](#)，选择项目管理，单击成员管理选择指定成员可修改该成员对应项目下单个应用的权限为只读/编辑/编辑

添加成员

×

成员名称

成员权限

所有功能权限

请输入应用名称

Q

▼

✓

应用名称

▼

✓

应用名称

☐ 只读

☒ 编辑

☐ 编辑并发布

✓

应用名称

☒ 只读

☐ 编辑

☐ 编辑并发布

已选择功能权限

▼

✓

应用名称

▼

✓

应用名称

☒ 只读

☐ 编辑

☐ 编辑并发布

✓

应用名称

☒ 只读

☐ 编辑

☐ 编辑并发布

确定

取消