ALIBABA CLOUD

阿里云

云服务总线 CSB

云服务总线CSB公共云合集

文档版本：20220420

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.产品计费

1.1. 价格说明

本文介绍CSB的计费模式及具体的价格说明。

开放平台采用后付费模式，按当天实际拥有的开放平台网关实例规格和节点数量计费，日结。

您可以在EDAS中创建CSB专享实例，并可以根据需要随时增、减实例的节点数量。CSB实例包含多种规格，不同的规格费用不同。

开放平台网关实例规格	单节点费用（元/天）
2核4G内存（不推荐在生产环境使用）	70
2核8G内存	105
4核8G内存	140

2.实例管理

2.1. 实例管理概述

实例创建完成后，可以对实例进行管理。

简介

CSB实例分为共享实例和专享实例，请参见实例。

> **说明** 共享实例仅用于体验试用，不建议正式生产使用。

创建实例

CSB基于实例提供各种功能，所以您首先需要创建实例，请参见创建实例。

管理实例

创建实例后，可以在EDAS控制台和CSB控制台管理实例。

- 在EDAS控制台主要是实例的集群资源操作，包括节点管理、暂停/恢复实例和删除实例，请参见管理服务开放实例。
- 在CSB控制台主要是实例上的业务操作。本文重点介绍在CSB控制台如何管理实例。

共享实例和专享实例的管理有所不同。

- 管理专享实例
  - 公开或隐藏实例
  - 为其他账号授权
  - 申请实例
  - 审批实例申请
  - 设置服务发布审批策略
- 管理共享实例：VPC地址管理
- 通用管理操作：管理数据源

2.2. 查看实例

您可以查看您拥有和获准使用的实例及实例详情。

查看实例列表

您可以在**实例列表**页面查看您拥有和获准使用实例。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面查看拥有和授权使用的实例。

实例信息说明：

- **实例地址**：默认提供RESTful服务和WebService服务访问地址。
  - 共享实例的服务访问地址是固定值，不可修改，请参见共享实例名称和服务访问地址。
  - 托管式专享实例的服务访问地址取决于创建实例时绑定的SLB，支持手动修改。您可在目标实例的**操作**列单击**更多**右侧的，在下拉列表中选择**服务访问地址管理**，然后根据需要手动修改服务访问地址。
- **状态**：包含**拥有**、**获准使用**和**审批中**。
  - **拥有**：您创建的实例。
  - **获准使用**：共享实例或者其他用户创建的，您申请并审批通过，可以使用的实例。
  - **审批中**：您已经申请使用，但在**审批中**的实例。
- **实例类别**：包含**托管式专享**和**共享**。
  - **托管式专享**：在EDAS中创建的CSB专享实例。
  - **共享**：共享实例，请参见实例。

> **说明** 共享实例仅用于体验试用，不建议正式生产使用。

查看实例详情

您可以在**实例概览**页面中查看该实例的详情，包含**服务发布**和**服务订阅**的相关数据。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击具体实例名称。

> **注意** 如果您使用共享实例，请参见共享实例信息使用CSB指定的共享实例，否则会导致发布失败。共享实例仅用于体验试用，不建议正式生产使用。

5. 在实例的**概览页面**查看该实例中**服务发布**和**服务订阅**相关数据。
   - **服务发布**：包含服务组数量、激活和停用的服务数量、不同时间周期（一天、一周和总计）的调用次数和错误次数。
   - **服务订阅**：包含订购数量、凭证数量、不同时间周期（一天、一周和总计）的调用次数和错误次数。

2.3. 隐藏实例

您创建的实例默认是公开的，其他账号可以搜索、申请该实例。您也可以根据实际情况隐藏实例，禁止其他账号搜索和申请。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面选择目标实例，在该实例的**操作**列关闭**可见**开关，隐藏该实例。

结果验证

关闭**可见**开关后，其他账号在**实例搜索**页面不能查看并申请该实例。

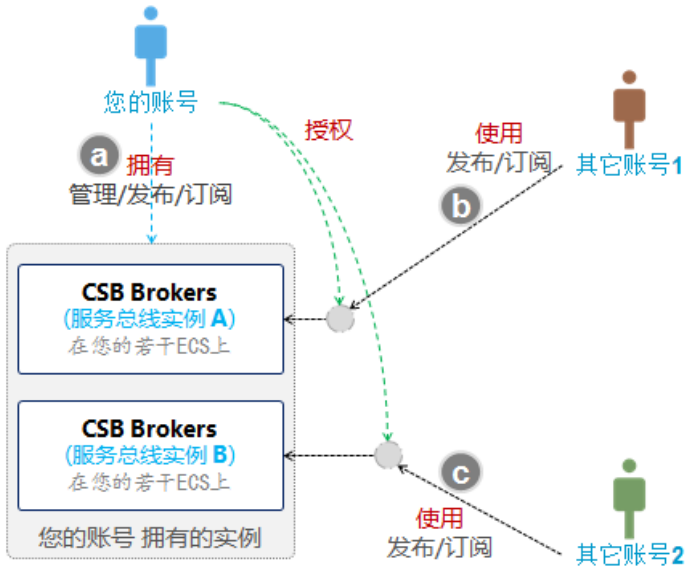
2.4. 为其他账号授权

您可以将您创建的实例授权给其他账号，使其具有该实例的使用权限。

授权简介

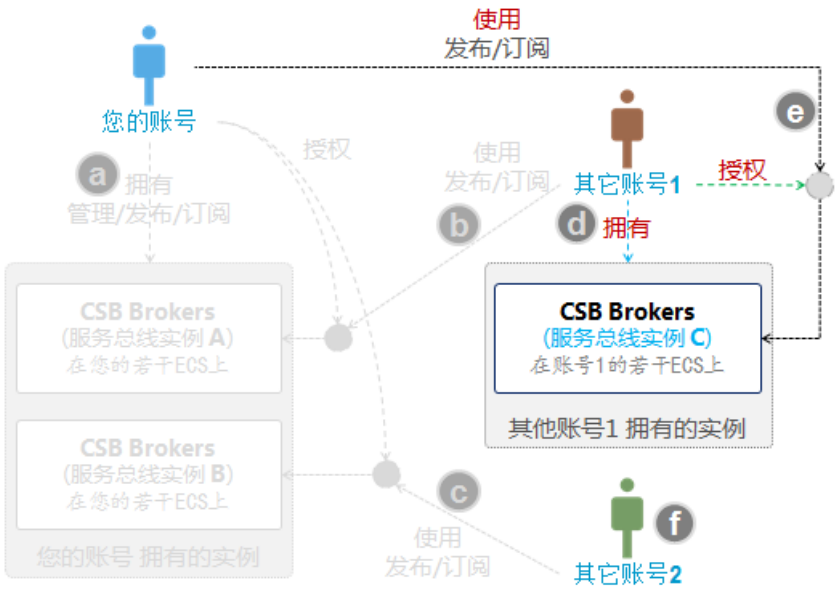
一个CSB实例就是一个单独的服务开放平台，被授权的账号可以在该实例上发布和订阅服务。

创建CSB实例的用户，即该实例的拥有者，可以为其他账号授权，使其能够使用该实例、在该实例上发布和订阅服务。



- 您拥有A、B两个实例。一般是两个用于不同目的和控制策略的开放平台，例如负责不同VPC内的服务对外开放；或者面对不同的消费方环境，如阿里云上的应用和外部第三方应用。
- 您授权*其他账号1*使用实例A，可以在其上发布或订阅服务。
- 您授权*其他账号2*使用实例B，可以在其上发布或订阅服务。

这是最简单的授权示例，而实际业务系统中很可能会更加复杂。



- *其他账号1*也拥有一个实例C。
- *其他账号1*授权您使用实例C，您可以在其上发布或订阅服务。
- *其他账号2*没有任何实例，只是使用您的实例B。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面选择目标实例，然后在该实例的**操作**列单击**更多**右侧的，在下拉列表中选择**授权用户**。
5. 在**授权用户**对话框中输入要授权的阿里云账号的**登录名**，单击**确认**。

结果验证

授权完成后，被授权的用户登录CSB控制台，并选择相同地域后，在**实例列表**页面可看到被授权使用的实例。

> **说明** 被授权的用户只能管理该实例的数据源和查看节点信息，不能进行其他管理操作。

2.5. 设置服务发布审批策略

您可以为您创建的专享实例上设置服务发布审批策略，实现对服务发布的直接管控。

前提条件

已创建专享实例，请参见创建共享实例。

> **注意** 只有该实例的创建者（拥有者）才能设置服务发布审批，被授权的用户无法设置。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面选择目标实例，然后在实例的**操作**列单击**更多**右侧的，在下拉列表中单击**服务发布审批设置**。
5. 在**服务发布审批设置**对话框中设置服务发布的审批规则，单击**确认**。
   - **不需要审批**：在该实例中发布服务不需要审批，直接处于激活状态。
   - **一级审批**：在该实例上发布的服务默认为待审批状态，需要指定的**一级审批人**审批后才处于激活状态。

> **说明** **一级审批人**可以是该实例的拥有者（创建该实例的用户）或使用者（被授权使用该实例的用户）。

   - **二级审批**：在该实例上发布的服务默认为待审批状态，需要经过指定的**一级审批人**和**二级审批人**审批后才处于激活状态。

> **说明** 一级审批人和二级审批人可以是该实例的拥有者（创建该实例的用户）或使用者（被授权使用该实例的用户），但不能为同一个用户。

2.6. 管理数据源

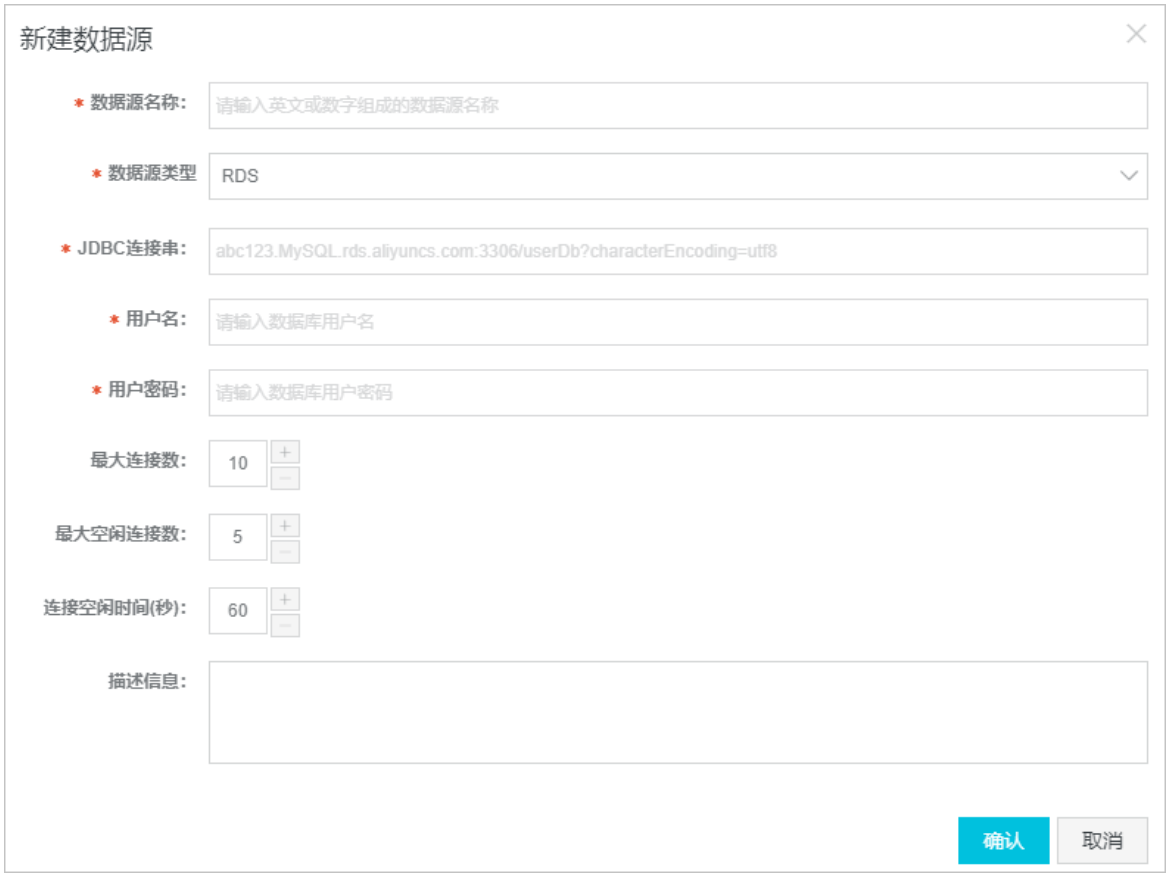
CSB支持通过JDBC（Java Database Connectivity）将后端数据库访问能力以RESTful方式发布为服务，所以在发布该类型服务时，需要管理CSB实例的数据源。

背景信息

- CSB目前支持MySQL兼容的数据库，包括RDS Mysql、DRDS、MySQL。
- CSB目前支持PostgreSQL兼容的数据库，包括RDS PostgreSQL、AnalyticDB PostgreSQL、PostgreSQL。
- 每个CSB实例可以配置多个数据源，多个服务可以共享一个数据源。

新建数据源

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面选择目标实例，然后在该实例的**操作**列单击**更多**右侧的，在下拉列表中单击**数据源管理**。
5. 在**数据源管理**对话框右上角单击**新建数据源**。
6. 在**新建数据源**设置数据源参数，单击**确认**。



新建数据源参数说明：

- **数据源名称**：只允许包含英文字母、数字、短划线（-）和下划线（_），长度为3~32个字符。
- **数据源类型**：包含*RDS*、*DRDS*、*MYSQL*和*PostgreSQL*，根据实际需求选择。
- **JDBC连接串**：格式为 `RDS域名` 、 `MySQL的IP:端口` 、 `PostgreSQL域名` 或 `PostgreSQL的IP:端口` 。
- **用户名**：登录数据库的用户名，最长32个字符。
- **用户密码**：登录数据库的密码，最长32个字符。
- **最大连接数**：范围为1~50。
- **最大空闲连接数**：范围为1~50。
- **连接空闲时间**：范围为5秒~600秒。
- **描述信息**：自定义数据源的标识信息。

结果验证

数据源新建完成后，在**数据源管理**对话框查看数据源信息与您新建的是否一致。

2.7. 申请实例

您可以申请某个实例的使用权限，申请通过后，即可在该实例上发布和订阅服务。

背景信息

需要申请的是其他用户创建的专享实例。共享实例无需申请，默认可以使用。

CSB实例默认处于公开模式，您可以在**实例搜索**页面中查看并申请其他用户创建的实例，在实例的拥有者审批通过后，即可在该实例上发布和订阅服务。

> **说明** 共享实例仅用于体验试用，不建议正式生产使用。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例搜索**。
4. 在**实例搜索**页面查找目标实例，在该实例的**操作**列单击**申请**。

   如果实例较多，也可以通过**实例名称**的关键字进行搜索。
5. 在**申请实例**对话框中输入**申请原因**，单击**确认**。

结果验证

- 申请提交后，在**我的申请**页面查看申请实例的**状态**是否为**待审批**。
- 申请成功后，在**我的申请**页面查看申请实例的**状态**是否为**已通过**。

2.8. 审批实例申请

您可以审批其他用户对您创建（拥有）的实例的使用申请。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**我的审批**。
4. 在**我的审批**页面，查看**申请人**、**实例名称**和**申请原因**。根据实际情况，在申请记录的**操作**列单击**通过**或**拒绝**，然后在弹出的确认对话框中单击**确认**。

结果验证

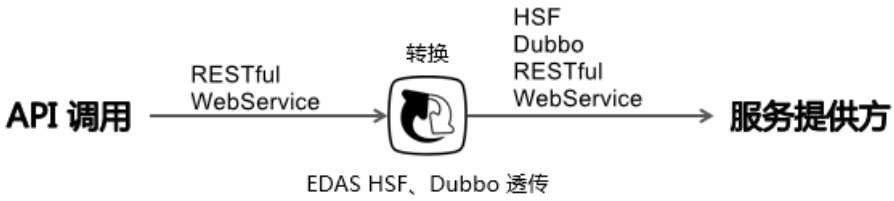
审批结束后，申请记录的**操作**列的按钮变为不可用（置灰）。

3.服务发布

3.1. 服务发布方式

CSB支持发布后端已有服务和导入Swagger API两种服务发布方式。当您发布后端已有服务时，可能会有不同的需求和场景。CSB提供了普通发布、路由发布和级联发布3种方式以满足您不同需求和场景。

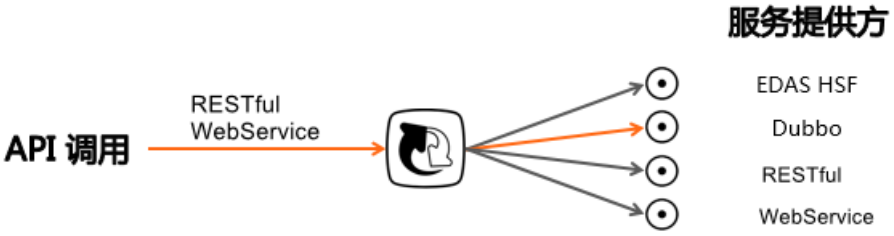
普通发布



后端只有一个服务接入时，可以使用普通发布在CSB上同时以相同或不同类型的接口协议开放。例如一个后端的HSF服务可以在CSB上同时开放成RESTful、WebService和EDAS HSF（透传）三个不同类型的服务入口。

目前，CSB后端接入RESTful、HSF、WebService、DUBBO类型的服务都可以开放成RESTful或者WebService，接入的SpringCloud、JDBC类型的服务只可以开放成RESTful。此外还支持Dubbo和EDAS HSF类型服务的透传开放。

路由发布



当后端有多个相同或不同类似服务接入时，可以使用路由发布。这些服务以同一个RESTful或WebService类型接口开放，被访问时，CSB会根据路由规则将请求转发到后端接入的不同服务。

此处的路由发布即内容路由发布，接入的请求根据参数值的不同，路由到多个不同的后端接入协议或者相同的接入协议不同的接入地址。

级联发布

针对复杂的多环境、多归属互通场景，云服务总线提供级联发布管理机制，即跨CSB实例的服务发布，也就是在一个CSB实例上接入已有服务，而在另一个CSB实例上发布出来，供订阅者消费。

级联链路可以跨2个或更多CSB实例，这些CSB实例可以归属不同的用户，甚至位于不同的CSB群组内。

当服务要发布的目标实例不是当前实例时，可以使用级联发布。级联发布会将该服务通过级联的方式发布到所选的目标实例中。这种场景下，服务的接入端和服务的发布开放端分布在不同的实例上，称之为源实例和目标实例。根据预先定义的级联发布规则，在源实例和目标实例之间还可能存在中间实例，形成一条级联链路。

> **说明** 级联发布的链路并不是随意的，需要群组管理员定义有方向的连通链路，指明链路中各个CSB实例的先后连接关系。例如CSB实例A上接入的服务通过实例B作为中转，最终在CSB实例C上开放，就构成了一条经由实例A到B到C的级联链路。级联服务经过的CSB实例在级联链路方向上需要进行实例间授信。级联链路的定义和管理，包括实例间授信操作，都由群组管理员，即CSB技术支持人员完成。

单实例发布与级联发布

- 如图中①所示，服务x发布在实例A上，被应用α（在实例A上订阅后）调用消费。
- 如图中②所示，服务y发布在实例B上，被应用β（在实例B上订阅后）调用消费。
- 如图中③所示，服务y在实例B接入，通过级联链路在实例A上发布，被应用α（在实例A上订阅后）通过实例A、B级联调用消费。需要预先定义实例B到A的级联链路，包括实例B对实例A的访问授信。

3.2. 发布服务

3.2.1. 发布后端已有服务

发布服务就是将一个已有的后端服务在某个CSB实例上注册，并以选定的一种或多种协议开放成API供消费者使用，同时对服务的消费做一定的访问控制。

前提条件

在体验发布服务前，您需要完成以下操作：

- 创建实例
- 新建服务组
- 了解服务发布的约束规范，请参见约束规范。

背景信息

服务发布包含以下三个概念。

- 接入服务：提供API对应的后端服务信息，让CSB能访问到这个已有的服务。
- 开放服务：指明API开放的协议，以及开放的接口和后端服务接口如何对应。
- 访问控制：指明API开放的策略，是否限流，对谁可见，访问是否需要授权。

CSB可以发布RESTful、SpringCloud、HSF、WebService、Dubbo和JDBC协议类型的服务。

> **说明** 本文仅介绍发布服务的整体流程和重要参数解释，如需详细了解流程和参数，请结合发布服务示例场景文档进行了解。具体信息，请参见发布服务示例场景概述。

发布服务流程

1. 进入发布服务页面
2. 命名服务
3. 设置接入协议
4. 设置开放协议
5. 设置访问限制
6. 发布服务

进入发布服务页面

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击具体实例名称。

> **注意** 如果您使用共享实例，请参见共享实例信息使用CSB指定的共享实例，否则会导致发布失败。共享实例仅用于体验试用，不建议正式生产使用。

5. 在**实例概览**页面左侧导航栏选择**发布者 > 发布服务**。
6. （可选）在**CSB服务版本**对话框中单击**新版发布（推荐）**。

> **说明** 该对话框仅在第一次发布服务出现，您可以选中**记住我的选择**。

命名服务

1. 在**路由服务发布**页面的**命名服务**页签中设置参数，单击**下一步**。

命名服务参数说明：

- **服务全名**：输入要发布的服务全名。
- **服务版本**：输入服务版本号。
- **所属服务组**：在下拉列表中选择该服务所属的服务组。在服务数量少时所属服务组可视作简单分类。
- **业务日志**：根据您的需要选择是否开启日志开关。

  开启日志开关后，会记录业务请求和响应的内容，您可在CSB Broker的*/home/admin/cloud-gateway/logs*的trace.log内查看日志详情。

  不建议长时间开启日志开关，避免日志存储占用过多内存，从而影响性能。
- **服务别名**：输入服务别名，用于标识该服务。
- **归属**：服务归属组织名、部门名称或者提供商名。
- **服务说明**：可以填写API的详细描述或者其它说明信息。最长128个汉字或字符。

> 说明
>
> - （**服务全名+服务版本**）在CSB实例上是唯一的。发布**服务全名**相同但**服务版本**不同的服务时，要确保在该实例下其他的相同**服务全名**的服务都已经处于**注销**状态。
> - 同一个**服务全名**的两个**服务版本**可以视为两个服务，即客户端调用的时候需要指定调用的**服务全名**和**服务版本**。

设置接入协议

1. 在**接入协议**页面选择**路由策略**。

   接入协议有两种路由策略：**直接路由**和**内容路由**。

   - **直接路由**：实际上是直接接入，不需要路由，只能选择一种接入协议。
   - **内容路由**：单击**新增**，在**编辑参数**页面设置路由条件，即定义Groovy条件规则，把接入的请求根据参数值的不同，路由到多个不同的后端接入协议或者相同的接入协议不同的接入地址。

格式为 `exportRequest.query.arg0 =='change to your value'` 。

- exportRequest表示开发入参。
- query.arg0表示URL中的参数arg0。
- change to your value表示参数值，HTTP入参支持query、body或header。

2. 在**接入协议**页面根据实际需求**选择一个接入协议**。

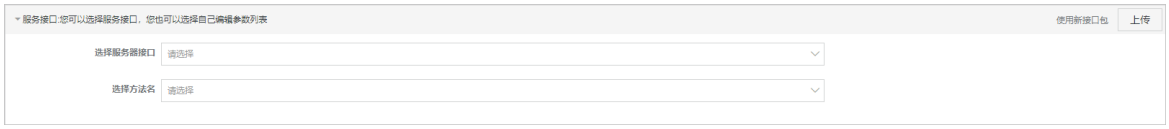
   CSB目前支持6种协议：**RESTful**、**SpringCloud**、**HSF**、**WebService**、**Dubbo**和**JDBC**。

3. 在**接入协议**页面根据选择的接入协议**配置接入服务**。

   CSB目前支持6种协议：**RESTful**、**SpringCloud**、**HSF**、**WebService**、**Dubbo**和**JDBC**。不同的协议，接入服务的参数有所不同。

   - **RESTful**：参数说明，请参见接入RESTful协议发布RESTful服务。
   - **SpringCloud**：参数说明，请参见如何发布和调用SpringCloud服务。
   - **HSF**：参数说明，请参见接入HSF协议发布RESTful服务。
   - **WebService**：在**选择一个接入协议**区域右上角单击**接入参数设置说明**，参考**接入参数设置说明**，根据您的Web服务的WSDL（Web Services Description Language）配置相关参数。
   - **Dubbo**：参数说明，请参见接入Dubbo协议发布RESTful服务。
   - **JDBC**：参数说明，请参见如何发布和调用JDBC数据服务。

4. 在**接入协议**页面设置**服务接口**。



   您可以选择服务接口，您也可以使用新接口包。

   手动的服务参数定义是一件繁琐的工作，CSB控制台提供了自动的参数加载功能，有两种方式进行参数的加载：

   - 使用命名服务步骤中选择的服务组所定义的接口文件。
   - 自定义的接口文件（它覆盖服务组中定义的接口文件）。

   这样就可以在定义参数的时候通过选择接口和对应的方法自动添加接口文件（Java类）中定义的参数。

5. 在**接入协议**页面**编辑入参**、**编辑出参**。

   根据接入协议，定义入参数和出参数，并设置参数的名称映射和参数顺序和层次。

   参数的名称映射是指发布入参数名为CSB开放的服务调用时候使用的参数名，发布出参数名为CSB接入后端服务的时候使用的参数名。参数的命名只能是数字，字母或下划线（_）。

   入参和出参请注意以下注意事项：

   - 参数的顺序要与接入服务要求的参数顺序一致。
   - 对于复杂的参数，需要定义参数的层次，来表示一个复杂的参数对象。

   每个参数都可以单独编辑，设置参数的属性信息：

   - **参数类型**： 支持下拉的基本参数类型和复杂的自定类型。
   - **扩展类型**： 参数的扩展类型，包含**正常**、**数组**、**列表**和**集合Set**， 例如方法 `hello(String arg0, String[] arg1, List&ltString> arg2)` 。
     - arg0为正常扩展类型（它可以是简单类型或者复杂类型）。

- arg1为数组扩展类型。
- arg2为列表扩展类型。

- **可选**：如果设置成参数可选，则调用时，发布的服务可以不输入该参数；如果是必选并且没有设置默认值，则调用时会报错。
- **传递方式**：包含*GET*和*POST*。
- **默认值**：如果不开放但又是后端服务必要的入参，需要指定默认值。
- **示例值**：生成接口文档时，提供参数的取值参考。

> **说明** 如果是透传的调用，可以不定义参数，发布的参数原样透传到后端的接入服务。

6. 接入协议设置完成后，单击**下一步**。

设置开放协议

1. 在**开放协议**页面选择**服务开放类型**。
   - 对于**直接路由**，开放协议是由所选择的接入协议决定。请参见协议转换种类。
   - 对于**内容路由**，开放协议目前只支持**RESTful**协议类型。
2. （可选）（适用于RESTful）在**开放协议**页面设置**开放Path**。
3. （可选）在**开放协议**页面的**编辑错误代码**右侧单击**添加错误码**。在弹出的**错误码编辑**对话框中输入数据的**错误代码**、**处置建议**和**错误说明**，然后单击**确认**。

> **说明** 此处的错误码为业务服务本身的错误码，CSB仅提供设置页面。调用服务时，返回设置好的错误码，以供服务调用者分析原因。

4. 在**开放协议**页面的**模拟返回结果**右侧单击**编辑模拟返回结果**，在弹出的**mock结果编辑**对话框中输入模拟返回结果，然后单击**保存**。

> **说明** 此处的返回结果为业务服务本身的返回结果，CSB仅提供设置页面。在调用服务请求中增加mock标识，可返回设置好的**模拟返回结果**。

5. 开放协议设置完成后，单击**下一步**。

设置访问限制

您可以设置发布的服务的**每秒最大调用量**，也可以打开或关闭**公开访问**开关。如果您关闭**公开访问**开关，则其他用户必须订阅您发布的服务才能使用该服务。

1. 在**限制访问**页面设置**每秒最大调用量**。

   每秒最大调用量用于指定该服务能接受的访问频度限制，即每秒最多调用次数。设置为*0*或者为空时代表不限制访问频度。
2. 在**限制访问**页面根据实际需求打开或关闭**公开访问**开关，设置使用该服务是否需要订阅。
3. （可选）在**限制访问**页面进行**熔断设置**。

> **说明** 降级对服务的**接入协议**、**服务开放类型**有所限制，请参见条件与约束。满足条件的服务还需要在**配置接入服务**时设置**超时时间（ms）**。

熔断参数配置说明：

- **启用熔断**：是否打开启用熔断开关。
- **熔断触发条件**：设置熔断触发的条件，例如*10*秒内服务请求数超过*100*次，且错误率大于*40%*。
- **HTTP状态码**：设置HTTP状态码。
- **启用降级**：是否打开启用降级开关。当启用熔断不启用降级时，在熔断状态下会抛出熔断异常。
- **降级逻辑**：输入降级逻辑代码。降级逻辑包含默认值和降级端点两种方式。
  - 如果您已有备用的端点，可配置降级端点。
  - 如果您没有备用的端点，可配置默认值，以保证服务调用可以正常进行。

  下面分别提供默认值和降级端点的配置示例。
  - 降级端点（和RESTful端点对应，熔断状态或原端点服务异常情况下调用降级端点的服务）示例：

    ```
    http://service.ken.com:8080/fallback/item/{name}
    ```

  - 默认值（返回调用方指定结果）示例（RESTful）：

    ```
    {
    "msg": "FALLBACK",
    "result": {
    "quantity": 9999
    },
    "code": "0",
    "innerMsg": "DefaultValue"
    }
    ```

    HSF或Dubbo方法返回值对应的JSON（JSON.toJSONString(result)）。

    对于HSF级联发布的服务，需要使用Hessian2序列化之后的字符串作为默认值，即调用 `Hessian2Encoder.encodeToString(result)` 获得的字符串。示例代码如下（需要先在*pom.xml*中添加Dubbo和Hessian的依赖）：

    ```
    <dependency>
        <groupId>org.apache.dubbo</groupId>
        <artifactId>dubbo</artifactId>
        <version>2.7.5</version>
    </dependency>
    <dependency>
        <groupId>com.taobao.hsf.hessian</groupId>
        <artifactId>hessian</artifactId>
        <version>4.0.7.bugfix12-tuning3</version>
    </dependency>
    ```

```
import org.apache.dubbo.common.io.UnsafeByteArrayInputStream;
import org.apache.dubbo.common.io.UnsafeByteArrayOutputStream;
import com.taobao.hsf.com.caucho.hessian.io.Hessian2Input;
import com.taobao.hsf.com.caucho.hessian.io.Hessian2Output;
import com.taobao.hsf.com.caucho.hessian.io.SerializerFactory;
public class Hessian2Encoder {
    private static final SerializerFactory factory = new com.taobao.hsf.com.caucho.
hessian.io.SerializerFactory();
    private static final String chars = "0123456789ABCDEF";
    public static byte[] encode(Object object) throws Exception {
        UnsafeByteArrayOutputStream byteArray = new UnsafeByteArrayOutputStream();
        Hessian2Output output = new Hessian2Output(byteArray);
        output.setSerializerFactory(factory);
        output.writeObject(object);
        output.close();
        byte[] bytes = byteArray.toByteArray();
        return bytes;
    }
    public static String encodeToString(Object object) throws Exception {
        byte[] bytes = encode(object);
        return bytesToHexString(bytes);
    }
    private static String bytesToHexString(byte[] bytes) {
        StringBuilder result = new StringBuilder(bytes.length);
        String s;
        for (int i = 0; i < bytes.length; i++) {
            s = Integer.toHexString(0xFF & bytes[i]);
            if (s.length() < 2) {
                result.append(0);
            }
            result.append(s.toUpperCase());
        }
        return result.toString();
    }
}
```

4. 限制访问设置完成后，单击**下一步**。

发布服务

1. 确认服务的基本信息、开放协议、接入协议、限制访问等设置是否无误。
2. 确认无误后，单击**完成发布** 。

3.2.2. 导入Swagger API

开放平台对满足OpenAPI 2.0规范的Swagger API提供快速创建服务的能力，并支持服务调用。本文介绍基于Swagger API发布服务和调用服务的流程。

前提条件

- 创建实例（如果使用共享实例，则不用创建实例。）
- 新建服务组

- 已存在Swagger API。本场景提供示例Swagger API，网址为*https://petstore.swagger.io/v2/swagger.json*。

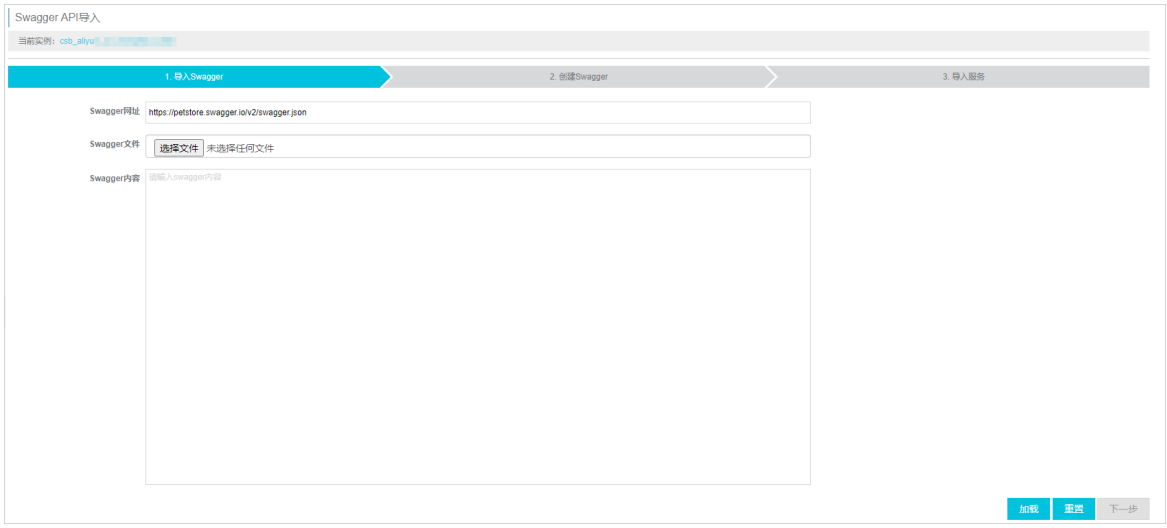
背景信息

本场景提供的示例Swagger API，存在于公网环境。因此介绍在共享实例上导入Swagger API发布服务和调用服务。

> **说明** 您的专享实例默认没有公网访问权限，如果您使用专享实例体验示例Swagger API发布和调用服务，请申请公网访问权限。当然您也可以使用您VPC内的Swagger API进行体验。

导入Swagger

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**单击具体实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 我的服务**。
6. 在**我的服务**页面单击**导入Swagger**。
7. 在**Swagger API导入**页面的**导入Swagger**页签中选择导入方式，然后单击**加载**。



导入Swagger参数说明。

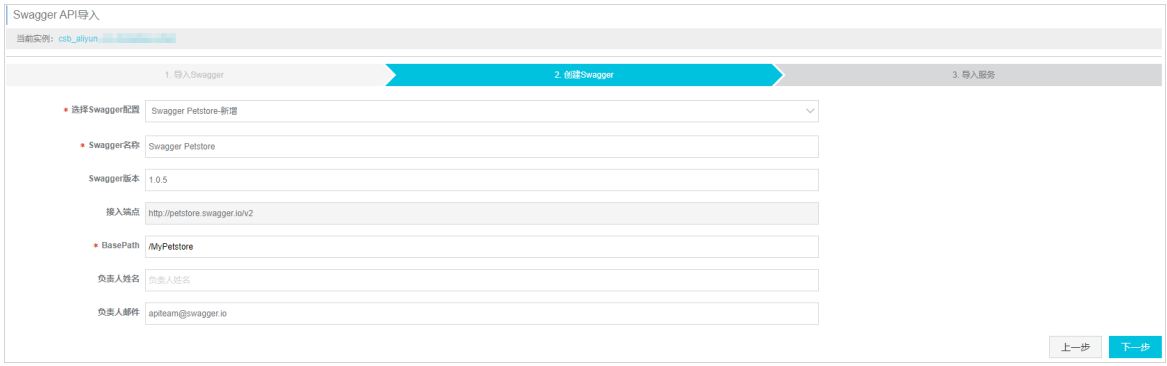
参数	描述	备注
Swagger网址	swagger.json的网址，如*https://petstore.swagger.io/v2/swagger.json*。	您只需选择其中任一种导入Swagger的方式。本场景以**Swagger网址**的方式为例。
Swagger文件	本地的Swagger文件。	

参数	描述	备注
Swagger内容	复制粘贴的Swagger API内容。 如果您选择**Swagger网址**或者**Swagger API**，在**加载**后也会在**Swagger内容**显示内容。	

8. 加载Swagger完成后，单击**下一步**。

创建Swagger

1. 在**创建Swagger**页签中设置Swagger参数，然后单击**下一步**。

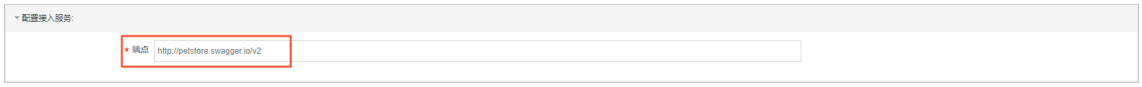


创建Swagger参数说明。

参数	描述
选择Swagger	选择Swagger的配置。 ○ 如果是首次加载Swagger API，该值为**Swagger Petstore-新增**。 ○ 后续加载Swagger API，可以选择之前的Swagger配置，该值为**Swagger Petstore（BasePath的值）**。
Swagger名称	加载的Swagger API存在**Swagger名称**字段，支持自定义设置。
Swagger版本	加载的Swagger API存在**Swagger版本**字段，支持自定义设置。
接入端点	加载Swagger API后自动生成，不支持设置。
BasePath	服务名称的前缀，支持自定义设置。
负责人姓名	自定义设置负责人名称。
负责人邮件	加载的Swagger API存在**负责人邮件**字段，此处支持自定义修改。

导入服务

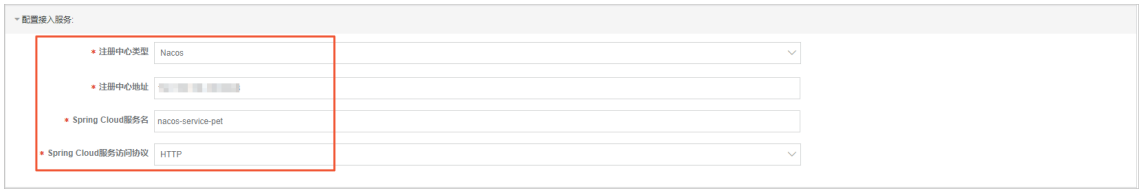
1. 设置接入协议。在基于Swagger API生成CSB服务场景中支持接入**RESTful**和**SpringCloud**协议。
   - 当选择接入协议为**RESTful**时，需要**配置接入服务**参数。



配置接入服务参数说明。

参数	描述
端点	RESTful服务地址。 加载Swagger API时默认生成**端点**值，您可根据需要修改**端点**值，修改后，对**发布服务**区域的所有服务生效。

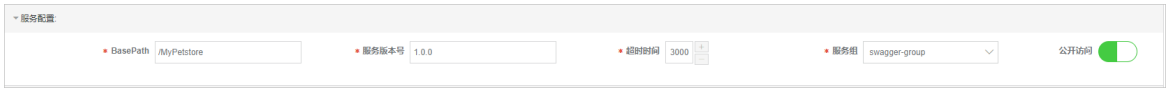
   - 当选择接入协议为**SpringCloud**时，需要设置**配置接入服务**参数。



配置接入服务参数说明。

参数	描述
注册中心类型	要发布的服务所注册的注册中心类型。目前支持**Nacos**、**Eureka**和**EDAS注册中心**。 请根据实际情况选择注册中心类型。
注册中心地址（适用于Nacos和Eureka注册中心）	注册中心的访问地址。
EDAS命名空间TID（适用于EDAS注册中心）	EDAS命名空间对应的TID。可以在EDAS控制台的**命名空间**页面查看。
Spring Cloud服务名	所选注册中心注册的Spring Cloud服务名称。 **说明** 由于注册中心可能在VPC内部，csbConsole无法直接访问，故CSB控制台上无法直接展示注册中心内已有的Spring Cloud服务列表。
Spring Cloud服务访问协议	根据实际情况选择服务访问协议，支持**HTTP**和**HTTPS**。

2. 在**服务配置**区域设置服务基础参数。



服务配置参数说明：

参数	描述
BasePath	服务名称的前缀，不可修改。
服务版本号	首次导入默认版本号**1.0.0**，后续导入时默认显示已存在的版本号。 您可根据实际需要修改**服务版本号**，修改后，对**发布服务**区域的所有服务生效。
超时时间	访问服务的超时时间，单位“ms”，默认值**3000**。
服务组	在下拉列表中选择已创建的服务组。 您可根据实际需要修改**服务组**，修改后，对**发布服务**区域的所有服务生效。
公开访问	是否允许公开访问，默认开启。 ○ 关闭公开访问时，非服务发布者需要订购服务并通过审批后，才可以调用服务。 ○ 开启公开访问时，无需订购服务，可直接调用服务。 您可根据实际需要修改**公开访问**的状态，修改后，对**发布服务**区域的所有服务生效。

3. 在**被占用服务**区域查看是否有被占用的服务。

   > **说明** 被占用的服务即是该服务在当前实例上，已经被其他用户发布，无法再次被发布。

4. 在**发布服务**区域对单个服务进行**编辑**、**不导入**或者**删除**。
   - 需要编辑单个服务的参数时，请单击目标服务后的**编辑**，设置完参数后单击**确认**。

编辑服务参数说明：

参数	描述
超时时间	服务的超时时间，单位“ms”，默认值**3000**。
QPS	每秒钟访问服务的量。默认值**0**，即不限制访问服务的量。
公开访问	是否允许公开访问，默认开启。 ■ 关闭公开访问时，非服务发布者需要订购服务并通过审批后，才可以调用服务。 ■ 开启公开访问时，即无需订购服务，可直接调用服务。

- 如果要不导入某个服务，请单击目标服务后的**不导入**，目标服务则会出现在**删除服务**区域。

  如果需要恢复服务，可在**删除服务**区域单击目标服务后的**还原**，目标服务则会出现在**发布服务**区域。

  > **注意** 当使用共享实例发布Swagger API服务时，同一个服务组下最大只能发布3个服务。因此您需要选择小于等于3个服务进行发布，其他服务请先**不导入**。待您完成第一批服务的调用后，在**我的服务**页面将其**注销**，然后重新选择服务发布，分批完成所有服务的发布与调用体验。

- 如果存在已经发布的服务，在该服务的操作列下有**删除**按钮。您可单击**删除**，该服务出现在**删除服务**区域。

  如果需要恢复服务，可在**删除服务**区域单击目标服务后的**还原**，目标服务则会出现在**发布服务**区域。

> **说明** 删除服务即是注销服务。

5. 在**导入服务**页签右下角单击**完成发布**。

   在基于Swagger API生成CSB服务场景中，发布的服务较多，发布时间较长，请耐心等待。

6. 在服务发布详情提示框单击**确认**。
   服务发布完成后，自动返回到**我的服务**页面，您可以查看所有服务信息。

调用服务

导入的Swagger API包含多个服务，本场景仅以调用**MyPetstore.findPetsByStatus**和**MyPetstore.getPetById**服务为例。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击目标实例的实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 我的服务**。
6. 在**我的服务**页面查找目标服务。

   本场景以调用**MyPetstore.findPetsByStatus**和**MyPetstore.getPetById**服务为例。

7. 调用**MyPetstore.findPetsByStatus**服务。
   i. 在**我的服务**页面单击**MyPetstore.findPetsByStatus**服务名称。
   ii. 在**服务详情**页面**服务基本信息**区域，查看并记录服务访问地址。

   本示例以展示RESTful服务访问地址为例：

   iii. 在**服务详情**页面**接入协议**区域，查看并记录入参信息。
   MyPetstore.findPetsByStatus服务的入参为status。status的取值信息请从Swagger API中获取，本示例的取值为*available*、*pending*和*sold*。

iv. 在浏览器地址栏输入服务访问地址，并在地址后加上入参取值，然后按enter键，界面返回服务详细信息。

服务访问地址为 `http://CSB服务地址:8086/服务版本/服务名称?入参取值` ，各参数说明如下：

- CSB服务地址即创建CSB实例时绑定的SLB的地址。
- 默认的访问端口为8086。
- 服务版本即该服务的版本号，如1.0.0。
- 服务名称在本示例中即是**MyPetstore.findPetsByStatus**。
- 入参取值即是服务的入参取值信息，如status=sold。

界面返回服务详细信息如下：

```
<pets>
    <Pet>
        <category>
            <id>8128823</id>
            <name>Category</name>
        </category>
        <id>6812466</id>
        <name>Bob</name>
        <photoUrls>
            <photoUrl>www.petPhotos.com/1.jpg</photoUrl>
            <photoUrl>www.petPhotos.com/2.jpg</photoUrl>
            <photoUrl>www.petPhotos.com/3.png</photoUrl>
        </photoUrls>
        <status>sold</status>
        <tags>
            <tag>
                <id>308662</id>
                <name>Tag1</name>
            </tag>
        </tags>
    </Pet>
</pets>
```

8. 调用**MyPetstore.getPetById**服务。

i. 在**我的服务**页面单击**MyPetstore.getPetById**服务名称。

ii. 在**服务详情**页面**服务基本信息**区域，查看并记录服务访问地址。

本示例以展示RESTful服务访问地址为例：

iii. 在**服务详情**页面**接入协议**区域，查看并记录入参信息。

MyPetstore.getPetById服务没有入参。

iv. 在浏览器地址栏输入服务访问地址，并把*petId*修改为真实值，然后按enter键，界面返回服务详细信息。

服务访问地址为 `http://CSB服务地址:8086/服务版本/服务名称/petId` ，各参数说明如下：

- CSB服务地址即创建CSB实例时绑定的SLB的地址。
- 默认的访问端口为8086。
- 服务版本即该服务的版本号，如1.0.0。
- 服务名称在本示例中即是**MyPetstore.getPetById**。
- petId是通过调用**MyPetstore.findPetsByStatus**服务查询出来的结果，如6812466。

界面返回服务详细信息如下：

```
<Pet>
    <category>
        <id>8128823</id>
        <name>Category</name>
    </category>
    <id>6812466</id>
    <name>Bob</name>
    <photoUrls>
        <photoUrl>www.petPhotos.com/1.jpg</photoUrl>
        <photoUrl>www.petPhotos.com/2.jpg</photoUrl>
        <photoUrl>www.petPhotos.com/3.png</photoUrl>
    </photoUrls>
    <status>sold</status>
    <tags>
        <tag>
            <id>308662</id>
            <name>Tag1</name>
        </tag>
    </tags>
</Pet>
```

3.2.3. 发布服务示例场景

3.2.3.1. 发布服务示例场景概述

发布服务就是将一个已有的后端服务在某个CSB实例上注册，并以选定的一种或多种协议开放成API供消费者使用，同时对服务的消费做一定的访问控制。

服务发布包含以下三个概念。：

- 接入服务：提供API对应的后端服务信息，让CSB能访问到这个已有的服务。
- 开放服务：指明API开放的协议，以及开放的接口和后端服务接口如何对应。
- 访问控制：指明API开放的策略，是否限流，对谁可见，访问是否需要授权。

CSB支持常用协议服务的接入和开放（RESTful/SpringCloud/HSF/WebService/Dubbo/JDBC），可扩展支持定制化的协议转换。

默认支持的服务接入、开放协议如下表所示：

支持的接入协议类型	对应支持的协议开放类型
RESTful	Restful、WebService
SpringCloud	Restful
HSF	Restful、WebService、HSF级联
WebService	Restful、WebService级联
Dubbo	Restful、WebService
JDBC	Restful

请根据您的真实场景选择接入协议和开放类型，本文提供了一些服务开放示例场景，仅供参考：

- 接入RESTful开放RESTful
- 接入HSF开放RESTful
- 接入WebService开放RESTful
- 接入RESTful开放WebService
- 接入HSF开放WebService
- 接入WebService开放WebService

3.2.3.2. 接入RESTful开放RESTful

本文以实际场景为例介绍接入RESTful协议开放RESTful协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入RESTful协议开放RESTful协议的服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

本文列举了接入RESTful协议开放RESTful协议的几种场景：

- 接入path开放query
- 接入query/body开放path
- 接入开放均为Path
- FORM
- JSON

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。

- curl

```
#使用Path传值
curl -H'Content-Type:x-www-form-urlencoded' --data "${Body参数}" "http://CSB服务地址:8086/${服务版本}/${服务名称}/开放Path?Query参数"
#使用query传值
curl --data "${Body参数}" "http://CSB服务地址:8086/开放Path?_api_name=${服务名称}&_api_version=${服务版本}&Query参数"
#使用Header传值
curl -H"_api_name:${服务名称}" -H"_api_version:${服务版本}" --data "${Body参数}" "http://CSB服务地址:8086/开放Path?Query参数"
```

- CSB SDK

```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

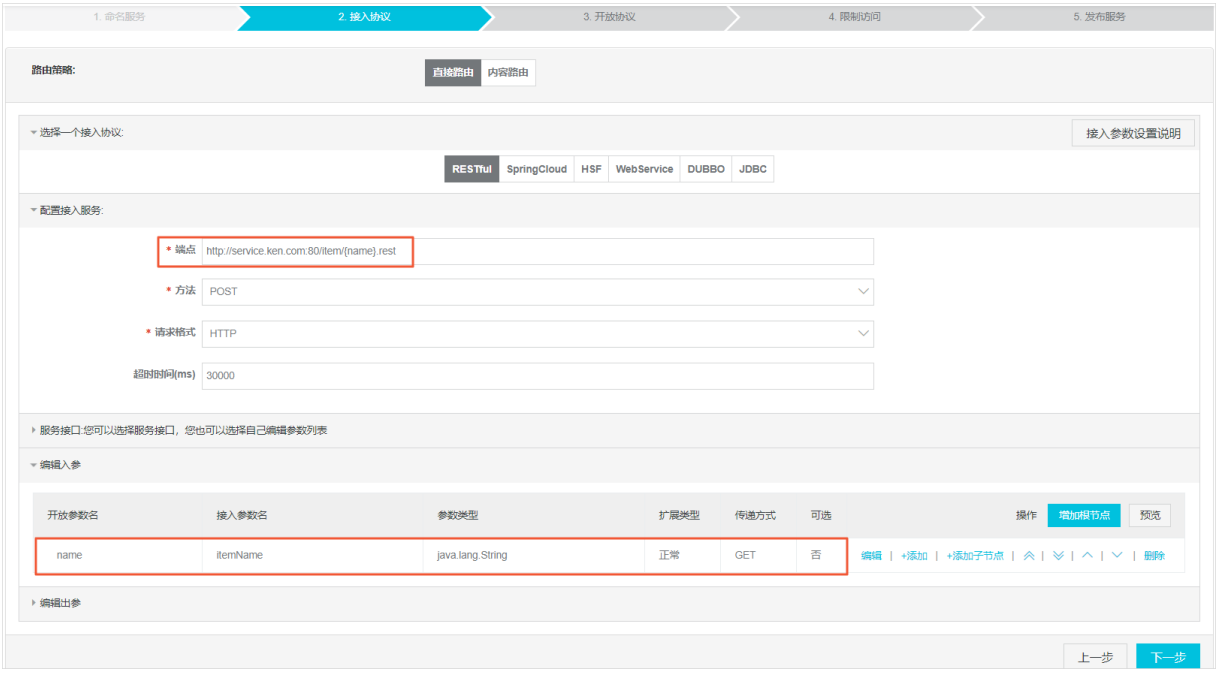
- 非公开访问（需要订购）的服务。

  在CSB SDK的基础上增加 `-ak '${访问凭证ak}' -sk '${访问凭证sk}'` 。

```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

接入path开放query

服务配置



调用服务

- 调用后端服务。

```
#请求后端Endpoint，其中的service.ken.com为供测试用的后端restful应用，benz对应PathVariable
curl http://service.ken.com/item/benz.rest
```

- 调用CSB开放的服务。

  您可以选择以下任一请求代码进行服务调用：

```
#通过Path参数curl CSB服务。
curl http://csb.target.server:8086/1.0.0/item.http.get-query?name=benz
```

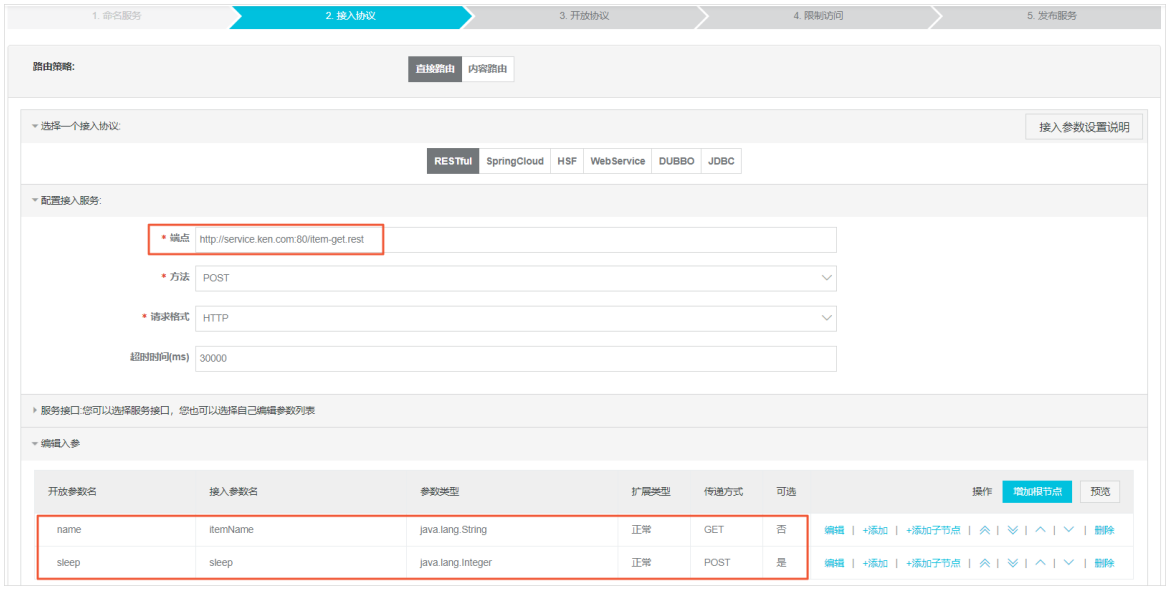
```
#通过Header参数curl CSB服务。
curl -H "_api_name:item.http.get-query" -H "_api_version:1.0.0" http://csb.target.server:8086/CSB?name=benz
```

```
#使用CSB SDK请求CSB服务。
java -jar http-client.jar -api item.http.get-query -version 1.0.0 -url http://csb.target.server:8086/CSB?name=benz
```

接入query/body开放path

服务配置

1. 在接入协议时设置端点，并设置对应的HTTP参数。



2. 设置开放协议的*path*。

接入后端代码

```
@RequestMapping(value = "/item-get", method = {RequestMethod.GET})
public ResultDTO<Item> get(@RequestParam("itemName") String itemName, @RequestParam("sleep"
) Integer sleep);
```

调用服务

- 调用后端服务。

  ```
  #请求后端Endpoint，其中的service.ken.com为供测试用的后端restful应用。
  curl "http://service.ken.com/item-get.rest?itemName=benz&sleep=10"
  ```

- 调用CSB开放服务。

  您可以选择以下任一请求代码进行服务调用：

  ```
  #通过Path参数curl CSB服务。
  curl http://csb.target.server:8086/1.0.0/item.http.item-get-path/benz/10
  ```

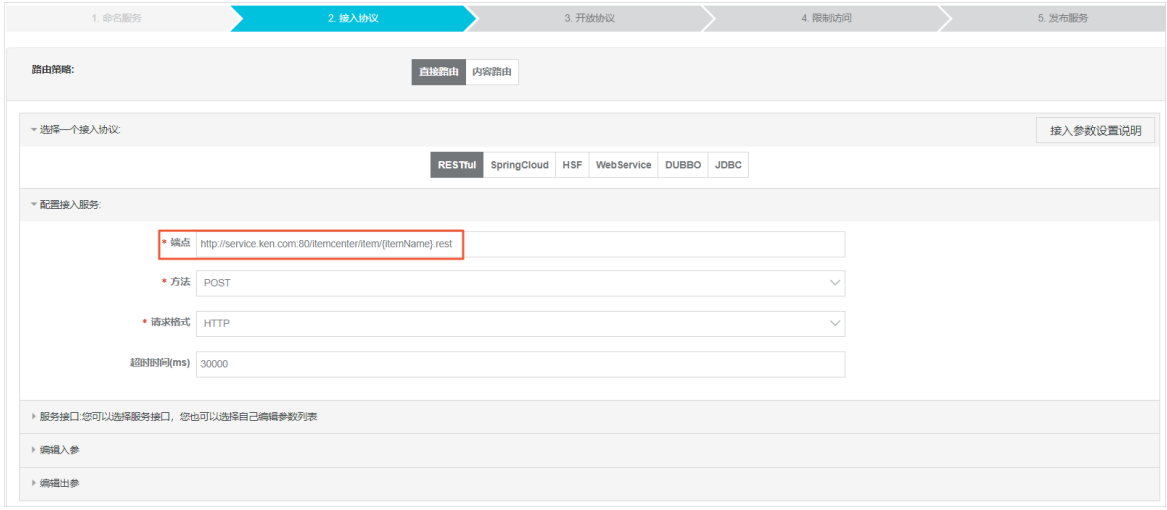
  ```
  #通过Header参数curl CSB服务。
  curl -H "_api_name:item.http.item-get-path" -H "_api_version:1.0.0" http://csb.target.s
  erver:8086/benz/10
  ```

  ```
  #使用CSB SDK请求CSB服务。
  java -jar http-client.jar -api item.http.item-get-path -version 1.0.0 -url http://csb.t
  arget.server:8086/benz/10
  ```

接入开放均为Path

服务配置

1. 在接入协议时的接入端点内设置*path*属性。



2. 设置开放协议的*path*属性。

接入后端代码

```
@GetMapping("/item/{itemName}")
public ResultDTO<Item> getByPath(@PathVariable("itemName") String itemName);
```

调用服务

- 调用后端服务。

```
#请求后端Endpoint，其中的service.ken.com为供测试用的后端restful应用。
curl http://service.ken.com/itemcenter/item/benz.rest
```

- 调用CSB开放服务。

  您可以选择以下任一请求代码进行服务调用：

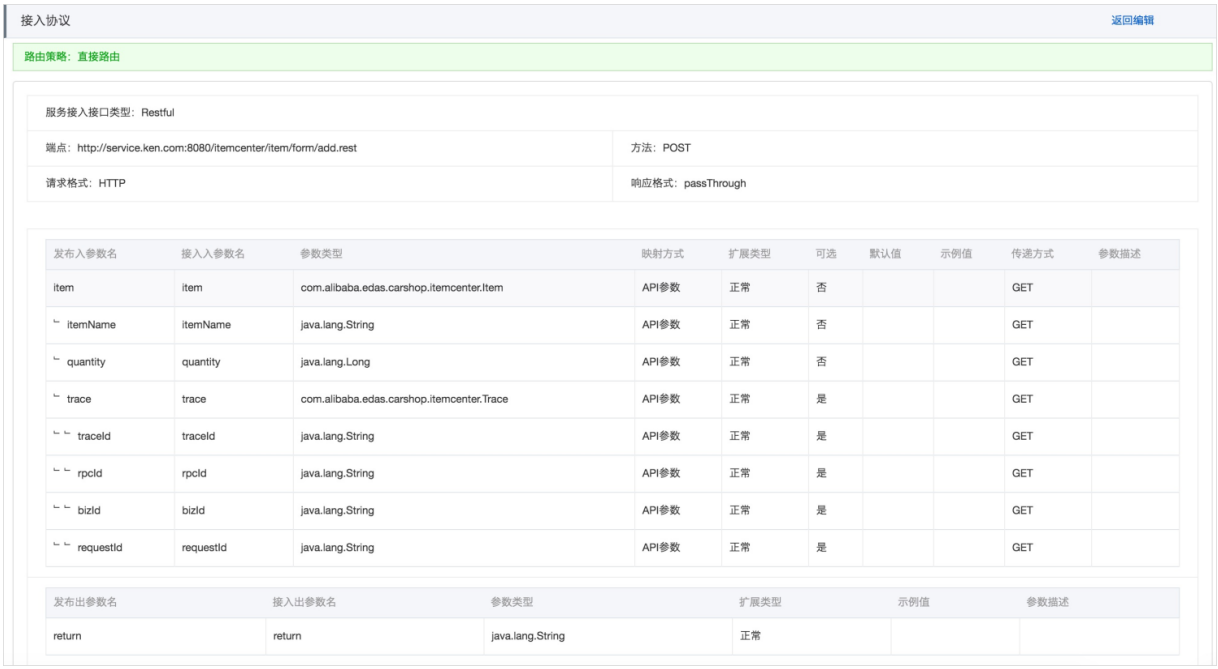
```
#通过Header参数curl CSB服务。
curl -H "_api_name:item.get" -H "_api_version:1.0.0" http://csb.target.server:8086/CSB/
benz
```

```
#使用CSB SDK请求CSB服务。
java -jar http-client.jar -api item.get -version 1.0.0 -url http://csb.target.server:80
86/CSB/benz
```

FORM

> **说明** FORM场景，接入RESTful协议支持开放为RESTful和WebService协议。

服务配置


接入协议 返回编辑

路由策略：直接路由

服务接入接口类型：Restful

端点：http://service.ken.com:8080/itemcenter/item/form/add.rest　方法：POST

请求格式：HTTP　响应格式：passThrough

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
item	item	com.alibaba.edas.carshop.itemcenter.Item	API参数	正常	否			GET	
itemName	itemName	java.lang.String	API参数	正常	否			GET	
quantity	quantity	java.lang.Long	API参数	正常	否			GET	
trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	API参数	正常	是			GET	
traceId	traceId	java.lang.String	API参数	正常	是			GET	
rpcId	rpcId	java.lang.String	API参数	正常	是			GET	
bizId	bizId	java.lang.String	API参数	正常	是			GET	
requestId	requestId	java.lang.String	API参数	正常	是			GET	

发布出参数名	接入出参数名	参数类型	扩展类型	示例值	参数描述
return	return	java.lang.String	正常		

接入后端服务

```
@PostMapping("/item/form/add")
public ResultDTO<Item> add(@JsonParam(value = "item") Item item);
```

调用服务

- 调用后端服务。

  ```
  #请求后端Endpoint，其中的service.ken.com为供测试用的后端restful应用。
  curl -X POST -d 'item={"itemName":"benz","quantity":10}' \
  http://service.ken.com:8080/itemcenter/item/form/add.rest
  ```

- 调用CSB开放服务。

  您可以选择以下任一请求代码进行服务调用：

  ```
  #通过Header参数curl CSB服务。
  curl -H "_api_name:item.form.add" -H "_api_version:1.0.0" \
  -X POST -d 'item={"itemName":"benz","quantity":10}' http://csb.target.server:8086/CSB
  ```

  ```
  #使用CSB SDK请求CSB服务。
  java -jar httpclient.jar -api item.form.add -version 1.0.0 -method post \
  -D 'item={"itemName":"benz","quantity":10}' -url http://csb.target.server:8086/CSB
  ```

请求结果

```
{
    "code": "0",
    "msg": "SUCCESS",
    "innerMsg": null,
    "result": {
        "trace": {
            "traceId": "0ba783c115667270553611002d****",
            "rpcId": "0",
            "requestId": "0ba783c115667270549241001d****"
        },
        "itemName": "benz",
        "quantity": 22627
    }
}
```

JSON

> **说明** JSON场景，接入RESTful协议只支持开放为RESTful协议。

服务配置

接入后端代码

```
@PostMapping("/item/add")
public ResultDTO<Item> add(@RequestBody Item item);
```

调用服务

- 调用后端服务。

  ```
  #请求后端Endpoint，其中的service.ken.com为供测试用的后端restful应用。
  curl -H "Content-Type:application/json" -X POST -d '{"itemName":"benz","quantity":10}' \
  http://service.ken.com:8080/itemcenter/item/add.rest
  ```

- 调用CSB开放服务。

  您可以选择以下任一请求代码进行服务调用：

  ```
  #通过Header参数curl CSB服务。
  curl -H "Content-Type:application/json" -H "_api_name:item.add" -H "_api_version:1.0.0"
  \
  -X POST -d '{"itemName":"benz","quantity":10}' http://csb.target.server:8086/CSB
  ```

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api item.add -version 1.0.0 \
-cbJSON '{"itemName":"benz","quantity":10}' -method post -url http://csb.target.server:
8086/CSB
```

请求结果

```
{
    "code": "0",
    "msg": "SUCCESS",
    "innerMsg": null,
    "result": {
        "trace": {
            "traceId": "0ba783c115667275786071002d****",
            "rpcId": "0",
            "requestId": "0ba783c115667275782281001d****"
        },
        "itemName": "benz",
        "quantity": 22677
    }
}
```

3.2.3.3. 接入HSF开放RESTful

本文以实际场景为例介绍接入HSF协议开放RESTful协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入HSF协议开放RESTful服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

本文列举了接入HSF协议开放RESTful协议的几种场景：

- FORM
- JSON
- 参数MAP（CSB服务和后端代码均未指定Map.Value类型）
- 参数List<Map>（CSB服务和后端代码均未指定Map.Value类型）
- 参数MAP（后端服务指定Map.Value类型）
- 参数POJO.Map

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。

- curl

```
#使用Path传值
curl -H'Content-Type:x-www-form-urlencoded' --data "${Body参数}" "http://CSB服务地址:8086/${服务版本}/${服务名称}/开放Path?Query参数"
#使用query传值
curl --data "${Body参数}" "http://CSB服务地址:8086/开放Path?_api_name=${服务名称}&_api_version=${服务版本}&Query参数"
#使用Header传值
curl -H"_api_name:${服务名称}" -H"_api_version:${服务版本}" --data "${Body参数}" "http://CSB服务地址:8086/开放Path?Query参数"
```

- CSB SDK

```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

- 非公开访问（需要订购）的服务。

  在CSB SDK的基础上增加 `-ak '${访问凭证ak}' -sk '${访问凭证sk}'` 。

```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

FORM

> **说明** FORM场景，接入HSF协议支持开放为RESTful和WebService协议。

服务配置

服务接入接口类型：HSF

接口全名：com.alibaba.edas.carshop.itemcenter.ItemService	方法名称：add
版本号：1.0.0	服务分组名称：HSF

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
item	item	com.alibaba.edas.carshop.itemcenter.Item	API参数	正常	是			GET	
└ itemName	itemName	java.lang.String	API参数	正常	是				
└ quantity	quantity	java.lang.Long	API参数	正常	是				
└ trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	API参数	正常	是			GET	
└ └ traceId	traceId	java.lang.String	API参数	正常	是			GET	
└ └ rpcId	rpcId	java.lang.String	API参数	正常	是			GET	
└ └ bizId	bizId	java.lang.String	API参数	正常	是			GET	
└ └ requestId	requestId	java.lang.String	API参数	正常	是			GET	

发布出参数名	接入出参数名	参数类型	扩展类型	示例值	参数描述
return	return	com.alibaba.edas.carshop.itemcenter.ResultDTO	正常		
└ code	code	java.lang.String	正常		
└ msg	msg	java.lang.String	正常		
└ innerMsg	innerMsg	java.lang.String	正常		
└ result	result	com.alibaba.edas.carshop.itemcenter.Item	正常		
└ └ itemName	itemName	java.lang.String	正常		
└ └ quantity	quantity	java.lang.Long	正常		
└ └ trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	正常		
└ └ └ traceId	traceId	java.lang.String	正常		
└ └ └ rpcId	rpcId	java.lang.String	正常		

接入后端代码

```
public ResultDTO<Item> add(Item item);
```

调用服务

您可以选择以下任一请求代码调用CSB开放的服务：

```
#通过Header参数curl CSB服务。
curl -H "_api_name:item.hsf.add" -H "_api_version:1.0.0" \
-d 'item={"itemName":"benz","quantity":10}' http://csb.target.server:8086/CSB
```

```
#使用CSB-SDK请求CSB服务
java -jar httpclient.jar -api item.hsf.add -version 1.0.0 -method post \
-D 'item={"itemName":"benz","quantity":10}' -url http://csb.target.server:8086/CSB
```

请求结果

```
{
    "body": {
        "msg": "SUCCESS",
        "result": {
            "itemName": "benz",
            "trace": {
                "traceId": "0ba783c115673295906521002d****",
                "requestId": "0ba783c115673295902731001d****",
                "rpcId": "0.1"
            },
            "quantity": 12
        },
        "code": "0"
    },
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115673295906521002d****"
}
```


JSON

> **说明** JSON场景，接入HSF协议只支持开放为RESTful协议；且还需要CSB Broker的版本大于等于3.8.1。

服务配置

服务接入接口类型：HSF

接口全名：com.alibaba.edas.carshop.itemcenter.ItemService	方法名称：add
版本号：1.0.0	服务分组名称：HSF

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
item	item	com.alibaba.edas.carshop.itemcenter.Item	API参数	正常	是			GET	
└ itemName	itemName	java.lang.String	API参数	正常	是				
└ quantity	quantity	java.lang.Long	API参数	正常	是				

发布出参数名	接入出参数名	参数类型	扩展类型	示例值	参数描述
return	return	com.alibaba.edas.carshop.itemcenter.ResultDTO	正常		
└ code	code	java.lang.String	正常		
└ msg	msg	java.lang.String	正常		
└ innerMsg	innerMsg	java.lang.String	正常		
└ result	result	com.alibaba.edas.carshop.itemcenter.Item	正常		
└ └ itemName	itemName	java.lang.String	正常		
└ └ quantity	quantity	java.lang.Long	正常		
└ └ trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	正常		
└ └ └ traceId	traceId	java.lang.String	正常		
└ └ └ rpcId	rpcId	java.lang.String	正常		
└ └ └ bizId	bizId	java.lang.String	正常		
└ └ └ requestId	requestId	java.lang.String	正常		

接入后端代码

```
public ResultDTO<Item> add(Item item);
```

调用服务

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api item.hsf.add.json -version 1.0.0 -method post \
-cbJSON '{"item":{"itemName":"benz","quantity":10}}' \
-url http://csb.target.server:8086/CSB
```

请求代码

```
{
    "body": {
        "msg": "SUCCESS",
        "result": {
            "itemName": "benz",
            "trace": {
                "traceId": "0ba783c115673295906521002d****",
                "requestId": "0ba783c115673295902731001d****",
                "rpcId": "0.1"
            },
            "quantity": 12
        },
        "code": "0"
    },
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115673295906521002d****"
}
```

参数MAP（CSB服务和后端代码均未指定Map.Value类型）

服务配置

接入后端代码

```
@Override
public <T> ResultDTO<List<T>> addMap(Map<String, T> param) {
    return ResultDTO.getSuccessResult(param.values().stream().collect(Collectors.toList()))
;
}
```

调用服务

> **说明** CSB服务和后端代码均未指定Map.Value类型，可通过JSON class转换为HSF对应参数；不指定class无法转换为指定T，此时T为JSONObject。

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api map.hsf.add -version 1.0.0 -method post \
-D 'param={
    "jack": {
       "name": "jack",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "江苏",
                "city": "南京"
            },
            "PresentAddress": {
                "province": "北京",
                "city": "北京"
            }
        },
        "class":"com.alibaba.edas.carshop.itemcenter.Contact"
    },
    "lucy": {
        "name": "lucy",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "河北",
                "city": "唐山"
            },
            "PresentAddress": {
                "province": "广东",
                "city": "深圳"
            }
        },
        "class":"com.alibaba.edas.carshop.itemcenter.Contact"
    }
}' -url http://localhost:8086/CSB
```

请求结果

```
{
    "body": {
        "msg": "SUCCESS",
        "result": [{
            "name": "lucy",
            "details": {
                "NativePlace": {
                    "province": "河北",
                    "city": "唐山",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "广东",
                    "city": "深圳",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }, {
            "name": "jack",
            "details": {
                "NativePlace": {
                    "province": "江苏",
                    "city": "南京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "北京",
                    "city": "北京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }],
        "code": "0"
    },
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115679524648981002d88dd"
}
```


参数List<Map>（CSB服务和后端代码均未指定Map.Value类型）

服务配置

接入后端代码

```
@Override
public <T> ResultDTO<List<T>> addMapList(List<Map<String, T>> params) {
    return ResultDTO.getSuccessResult(params.stream().map(x -> x.values()).flatMap(x -> x.s
tream()).collect(Collectors.toList()));
}
```

调用服务

> **说明** CSB服务和后端代码均未指定Map.Value类型，可通过JSON class转换为HSF对应参数；不指定class无法转换为指定T，此时T为JSONObject。

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api map.hsf.adds -version 1.0.0 -method post \
-D 'params=[{
    "jack": {
        "name": "jack",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "江苏",
                "city": "南京"
            },
            "PresentAddress": {
                "province": "北京",
                "city": "北京"
            }
        },
        "class":"com.alibaba.edas.carshop.itemcenter.Contact"
    },
    "lucy": {
        "name": "lucy",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "河北",
                "city": "唐山"
            },
            "PresentAddress": {
                "province": "广东",
                "city": "深圳"
            }
        },
        "class":"com.alibaba.edas.carshop.itemcenter.Contact"
    }
}, {
    "lily": {
        "name": "lily",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "江苏",
                "city": "南京"
            },
            "PresentAddress": {
                "province": "北京",
                "city": "北京"
            }
        },
        "class":"com.alibaba.edas.carshop.itemcenter.Contact"
    }
}]' -url http://localhost:8086/CSB
```

请求结果

```
{
```

```
{
    "body": {
        "msg": "SUCCESS",
        "result": [{
            "name": "lucy",
            "details": {
                "NativePlace": {
                    "province": "河北",
                    "city": "唐山",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "广东",
                    "city": "深圳",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }, {
            "name": "jack",
            "details": {
                "NativePlace": {
                    "province": "江苏",
                    "city": "南京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "北京",
                    "city": "北京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }, {
            "name": "lily",
            "details": {
                "NativePlace": {
                    "province": "江苏",
                    "city": "南京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "北京",
                    "city": "北京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }],
        "code": "0"
    },
    "code": 200
```

```
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115679526561641002d8d2f"
}
```

参数MAP（后端服务指定Map.Value类型）

服务配置

接入后端代码

指定Map.Value类型。

```
@Override
public ResultDTO<List<Contact>> addMapFixedValType(Map<String, Contact> param) {
    return ResultDTO.getSuccessResult(Lists.newArrayList(param.values()));
}
```

调用服务

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api map.hsf.add.fixtype -version 1.0.0 -method post \
-D 'param={
    "jack": {
        "name": "jack",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "江苏",
                "city": "南京"
            },
            "PresentAddress": {
                "province": "北京",
                "city": "北京"
            }
        }
    },
    "lucy": {
        "name": "lucy",
        "age": 10,
        "details": {
            "NativePlace": {
                "province": "山西",
                "city": "太原"
            },
            "PresentAddress": {
                "province": "浙江",
                "city": "杭州"
            }
        }
    }
}' -url http://localhost:8086/CSB
```

请求结果

```
{
    "body": {
        "msg": "SUCCESS",
        "result": [{
            "name": "lucy",
            "details": {
                "NativePlace": {
                    "province": "山西",
                    "city": "太原",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "浙江",
                    "city": "杭州",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }, {
            "name": "jack",
            "details": {
                "NativePlace": {
                    "province": "江苏",
                    "city": "南京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "北京",
                    "city": "北京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        }],
        "code": "0"
    },
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115679539733771002dabe8"
}
```


参数POJO.Map

服务配置

服务接入接口类型：HSF

接口全名：com.alibaba.edas.carshop.itemcenter.ItemService	方法名称：addMapField
版本号：1.0.0	服务分组名称：HSF

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
contact	contact	com.alibaba.edas.carshop.itemcenter.Contact	API参数	正常	是			GET	
└ name	name	java.lang.String	API参数	正常	是				
└ age	age	int	API参数	正常	是				
└ details	details	java.util.Map	API参数	正常	是				

接入后端代码

```
package com.alibaba.edas.carshop.itemcenter;
import java.util.Map;
@Data
public class Contact extends JsonClass {
    private String name;
    private int age;
    private Map<String, Detail> details;
}
@Data
class Detail extends JsonClass {
    private String province;
    private String city;
    private String area;
    private String address;
}
public ResultDTO<Contact> addMapField(Contact contact) {
    return ResultDTO.getSuccessResult(contact);
}
```

调用服务

```
#使用CSB SDK请求CSB服务。
java -jar httpclient.jar -api map-field.hsf.add -version 1.0.0 -method post \
-D 'contact={
    "name": "jack",
    "age": 10,
    "details": {
        "NativePlace": {
            "province": "江苏",
            "city": "南京"
        },
        "PresentAddress": {
            "province": "北京",
            "city": "北京"
        }
    }
}' -url http://localhost:8086/CSB
```

请求结果

```
{
    "body": {
        "msg": "SUCCESS",
        "result": {
            "name": "jack",
            "details": {
                "NativePlace": {
                    "province": "江苏",
                    "city": "南京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                },
                "PresentAddress": {
                    "province": "北京",
                    "city": "北京",
                    "class": "com.alibaba.edas.carshop.itemcenter.Detail"
                }
            },
            "class": "com.alibaba.edas.carshop.itemcenter.Contact",
            "age": 10
        },
        "code": "0"
    },
    "code": 200,
    "message": "SUCCESS",
    "requestId": "0ba783c115679538866181002da9ae"
}
```


3.2.3.4. 接入WebService开放RESTful

本文以实际场景为例介绍接入WebService协议开放RESTful协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入WebService协议开放RESTful服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

本文列举了接入WebService协议开放RESTful协议的几种场景：

- FORM
- FORM复杂

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。

- curl

```
#使用Path传值
curl -H'Content-Type:x-www-form-urlencoded' --data "${Body参数}" "http://CSB服务地址:8086/${服务版本}/${服务名称}/开放Path?Query参数"
#使用query传值
curl --data "${Body参数}" "http://CSB服务地址:8086/开放Path?_api_name=${服务名称}&_api_version=${服务版本}&Query参数"
#使用Header传值
curl -H"_api_name:${服务名称}" -H"_api_version:${服务版本}" --data "${Body参数}" "http://CSB服务地址:8086/开放Path?Query参数"
```

- CSB SDK

```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

- 非公开访问（需要订购）的服务。

  在CSB SDK的基础上增加 `-ak '${访问凭证ak}' -sk '${访问凭证sk}'` 。

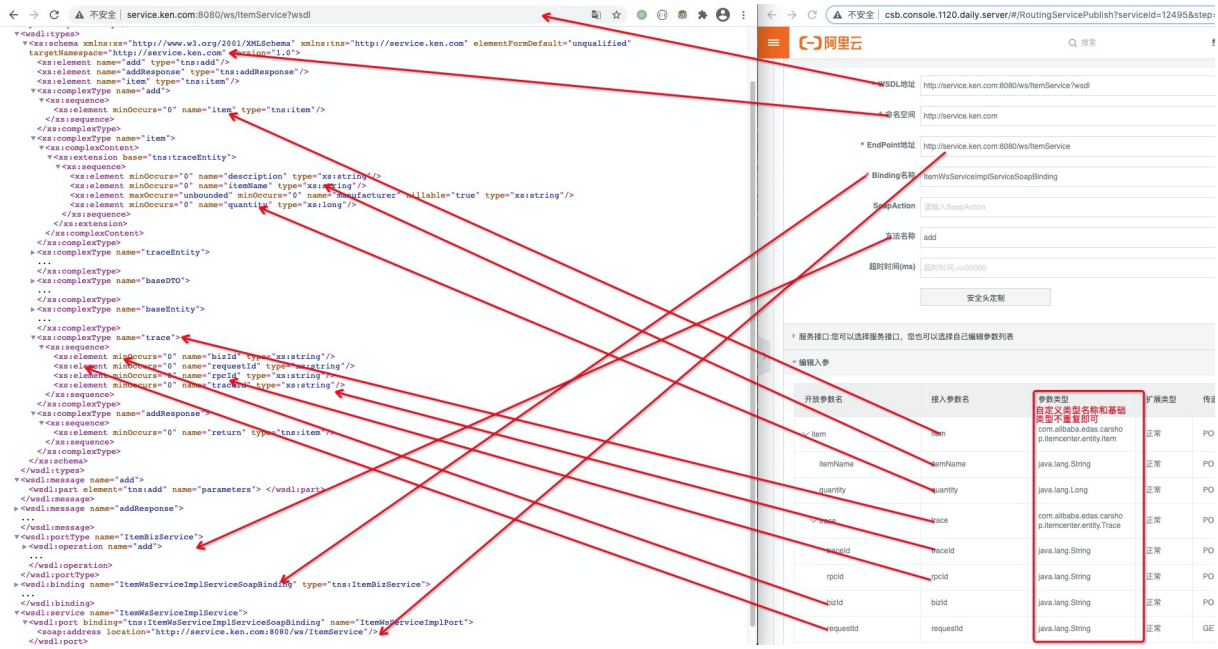
```
#非JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -D'param1=value1' -D'param2=value2' -url http://CSB服务地址:8086/开放Path
#JSON请求。
java -jar http-client.jar -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} -cbJSON "${JSONBody}" -url http://CSB服务地址:8086/开放Path
```

FORM

> **说明** FORM场景，接入WebService协议只支持开放为RESTful协议。

服务配置

接入WebService开放RESTful，不支持出参配置。



接入后端服务

```
@WebService(endpointInterface = "com.alibaba.edas.carshop.itemcenter.biz.ItemBizService", t
argetNamespace = "http://service.ken.com")
@BindingType(value = SOAPBinding.SOAP12HTTP_BINDING)
public interface ItemBizService {
    Item add(@WebParam(name = "item") Item item);
}
```

调用服务

```
#使用CSB-SDK请求CSB开放服务。
java -jar httpclient.jar -api item.ws.add -version 1.0.0 -method post \
-D 'item={"itemName":"benz","quantity":10}' -url http://csb.target.server:8086/CSB
#通过Header参数curl CSB服务
curl -H "_api_name:multilevel" -H "_api_version:1.0.0" -X POST \
-d 'item={"itemName":"benz","quantity":10}' http://csb.target.server:8086/CSB
```

请求结果

```
{
    "ns2:addResponse": {
        "return": {
            "itemName": "benz",
            "quantity": 20
        }
    }
}
```

WSDL

```
<wsdl:definitions
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://service.ken.com"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="ItemWsServiceImplService" target
Namespace="http://service.ken.com">
    <wsdl:types>
        <xs:schema
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://service.ken.com" elementFormDefault="unqualified" targetNames
pace="http://service.ken.com" version="1.0">
            <xs:element name="add" type="tns:add"/>
            <xs:element name="addResponse" type="tns:addResponse"/>
            <xs:element name="item" type="tns:item"/>
            <xs:complexType name="add">
                <xs:sequence>
                    <xs:element minOccurs="0" name="item" type="tns:item"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="item">
                <xs:complexContent>
                    <xs:extension base="tns:traceEntity">
                        <xs:sequence>
                            <xs:element minOccurs="0" name="description" type="xs:string"/>
                            <xs:element minOccurs="0" name="itemName" type="xs:string"/>
                            <xs:element maxOccurs="unbounded" minOccurs="0" name="manufactu
rer" nillable="true" type="xs:string"/>
                            <xs:element minOccurs="0" name="quantity" type="xs:long"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="traceEntity">
                <xs:complexContent>
                    <xs:extension base="tns:baseDTO">
                        <xs:sequence>
                            <xs:element minOccurs="0" name="trace" type="tns:trace"/>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="baseDTO">
                <xs:complexContent>
                    <xs:extension base="tns:baseEntity">
                        <xs:sequence>
                            <xs:element minOccurs="0" name="gmtCreated" type="xs:dateTime"/
>
                        </xs:sequence>
                    </xs:extension>
                </xs:complexContent>
            </xs:complexType>
            <xs:complexType name="baseEntity">
                <xs:sequence>
                    <xs:element minOccurs="0" name="id" type="xs:anyType"/>
                </xs:sequence>
            </xs:complexType>
```

```
            </xs:complexType>
            <xs:complexType name="trace">
                <xs:sequence>
                    <xs:element minOccurs="0" name="bizId" type="xs:string"/>
                    <xs:element minOccurs="0" name="requestId" type="xs:string"/>
                    <xs:element minOccurs="0" name="rpcId" type="xs:string"/>
                    <xs:element minOccurs="0" name="traceId" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="addResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:item"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="add">
        <wsdl:part element="tns:add" name="parameters"></wsdl:part>
    </wsdl:message>
    <wsdl:message name="addResponse">
        <wsdl:part element="tns:addResponse" name="parameters"></wsdl:part>
    </wsdl:message>
    <wsdl:portType name="ItemBizService">
        <wsdl:operation name="add">
            <wsdl:input message="tns:add" name="add"></wsdl:input>
            <wsdl:output message="tns:addResponse" name="addResponse"></wsdl:output>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="ItemWsServiceImplServiceSoapBinding" type="tns:ItemBizService">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="add">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="add">
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="addResponse">
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ItemWsServiceImplService">
        <wsdl:port binding="tns:ItemWsServiceImplServiceSoapBinding" name="ItemWsServiceImp
lPort">
            <soap:address location="http://service.ken.com:8080/ws/ItemService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

请求后端服务

代码的 `service.ken.com` 是测试用的后端RESTful应用。

```
curl --location --request POST 'http://service.ken.com:8080/ws/ItemService' \
--header 'Content-Type: application/xml' \
--data-raw '<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wst="http://service.ken.com">
    <soapenv:Header/>
    <soapenv:Body>
        <wst:add>
            <item>
                <itemName>benz</itemName>
                <quantity>10</quantity>
            </item>
        </wst:add>
    </soapenv:Body>
</soapenv:Envelope>'
```

后端服务返回

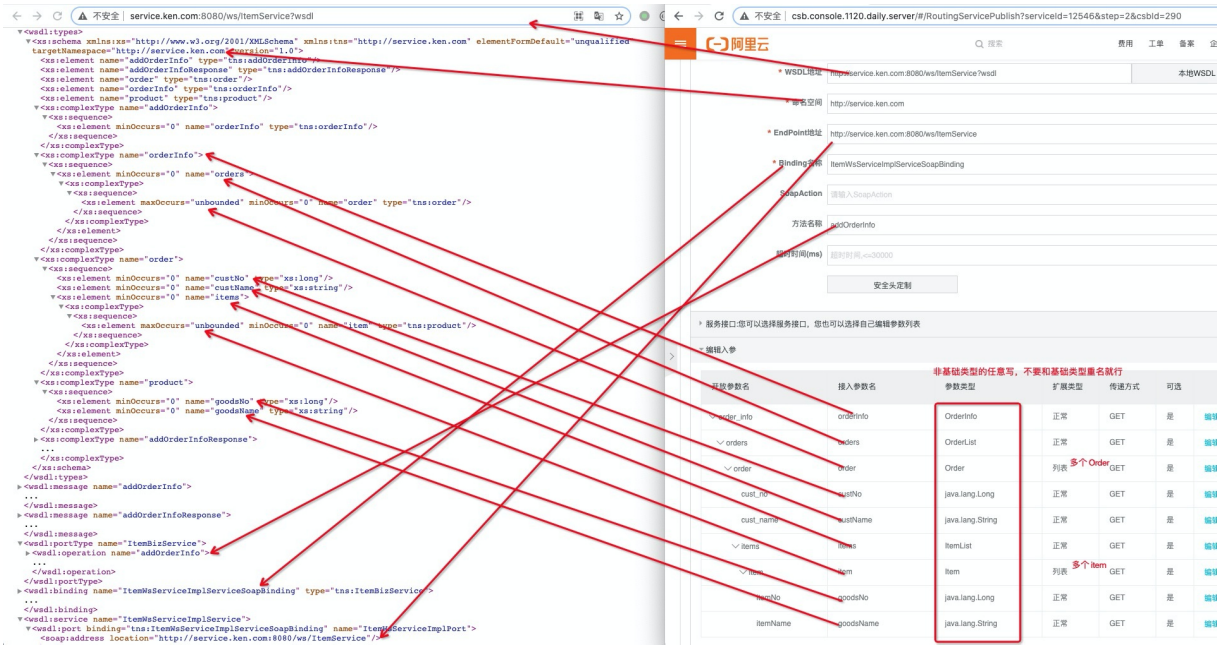
```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:addResponse
            xmlns:ns2="http://service.ken.com">
            <return>
                <itemName>benz</itemName>
                <quantity>20</quantity>
            </return>
        </ns2:addResponse>
    </soap:Body>
</soap:Envelope>
```

FORM复杂

> **说明** FORM复杂场景，接入WebService协议只支持开放为RESTful协议。

服务配置

接入WebService开放RESTful，不支持出参配置。



接入后端服务

```
@WebService(endpointInterface = "com.alibaba.edas.carshop.itemcenter.biz.ItemBizService", t
argetNamespace = "http://service.ken.com")
@BindingType(value = SOAPBinding.SOAP12HTTP_BINDING)
public interface ItemBizService {
    OrderInfo addOrderInfo(@WebParam(name = "orderInfo") OrderInfo orderInfo);
}
```

请求代码

```
#通过CSB SDK请求CSB服务
java -jar httpclient.jar -api multilevel -version 1.0.0 -method post \
-D 'orderInfo={
    "orders": {
        "order": [{
            "custName": "doctest",
            "custNo": 119640,
            "items": {
                "item": [{
                    "goodsName": "cookies",
                    "goodsNo": 1
                }, {
                    "goodsName": "apple",
                    "goodsNo": 2
                }]
            }
        }, {
            "custName": "apptest",
            "custNo": 87981,
            "items": {
                "item": [{
                    "goodsName": "banana",
                    "goodsNo": 3
```

```
                }, {
                    "goodsName": "coffee",
                    "goodsNo": 4
                }]
            }
        }]
    }
}' -url http://csb.target.server:8086/CSB
#通过Header参数curl CSB服务
curl -H "_api_name:multilevel" -H "_api_version:1.0.0" -X POST -d 'orderInfo={
    "orders": {
        "order": [{
            "custName": "doctest",
            "custNo": 119640,
            "items": {
                "item": [{
                    "goodsName": "cookies",
                    "goodsNo": 1
                }, {
                    "goodsName": "apple",
                    "goodsNo": 2
                }]
            }
        }, {
            "custName": "apptest",
            "custNo": 87981,
            "items": {
                "item": [{
                    "goodsName": "banana",
                    "goodsNo": 3
                }, {
                    "goodsName": "coffee",
                    "goodsNo": 4
                }]
            }
        }]
    }
}' http://csb.target.server:8086/CSB
```

请求结果

```
{
    "ns2:addOrderInfoResponse": {
        "return": {
            "orders": {
                "order": [{
                    "custNo": 119640,
                    "custName": "doctest",
                    "items": {
                        "item": [{
                            "goodsNo": 1,
                            "goodsName": "cookies"
                        }, {
                            "goodsNo": 2,
                            "goodsName": "apple"
                        }]
                    }
                }, {
                    "custNo": 87981,
                    "custName": "apptest",
                    "items": {
                        "item": [{
                            "goodsNo": 3,
                            "goodsName": "banana"
                        }, {
                            "goodsNo": 4,
                            "goodsName": "coffee"
                        }]
                    }
                }]
            }
        }
    }
}
```

WSDL

```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://service.ken.com"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="ItemWsServiceImplService" target
Namespace="http://service.ken.com">
    <wsdl:types>
        <xs:schema
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns:tns="http://service.ken.com" elementFormDefault="unqualified" targetNames
pace="http://service.ken.com" version="1.0">
            <xs:element name="addOrderInfo" type="tns:addOrderInfo"/>
            <xs:element name="addOrderInfoResponse" type="tns:addOrderInfoResponse"/>
            <xs:element name="order" type="tns:order"/>
            <xs:element name="orderInfo" type="tns:orderInfo"/>
```

```
            <xs:element name="product" type="tns:product"/>
            <xs:complexType name="addOrderInfo">
                <xs:sequence>
                    <xs:element minOccurs="0" name="orderInfo" type="tns:orderInfo"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="orderInfo">
                <xs:sequence>
                    <xs:element minOccurs="0" name="orders">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element maxOccurs="unbounded" minOccurs="0" name="order
" type="tns:order"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="order">
                <xs:sequence>
                    <xs:element minOccurs="0" name="custNo" type="xs:long"/>
                    <xs:element minOccurs="0" name="custName" type="xs:string"/>
                    <xs:element minOccurs="0" name="items">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element maxOccurs="unbounded" minOccurs="0" name="item"
type="tns:product"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="product">
                <xs:sequence>
                    <xs:element minOccurs="0" name="goodsNo" type="xs:long"/>
                    <xs:element minOccurs="0" name="goodsName" type="xs:string"/>
                </xs:sequence>
            </xs:complexType>
            <xs:complexType name="addOrderInfoResponse">
                <xs:sequence>
                    <xs:element minOccurs="0" name="return" type="tns:orderInfo"/>
                </xs:sequence>
            </xs:complexType>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="addOrderInfo">
        <wsdl:part element="tns:addOrderInfo" name="parameters"></wsdl:part>
    </wsdl:message>
    <wsdl:message name="addOrderInfoResponse">
        <wsdl:part element="tns:addOrderInfoResponse" name="parameters"></wsdl:part>
    </wsdl:message>
    <wsdl:portType name="ItemBizService">
        <wsdl:operation name="addOrderInfo">
            <wsdl:input message="tns:addOrderInfo" name="addOrderInfo"></wsdl:input>
            <wsdl:output message="tns:addOrderInfoResponse" name="addOrderInfoResponse"></w
```

```
            <wsdl:output message="tns:addOrderInfoResponse" name="addOrderInfoResponse"></w
sdl:output>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="ItemWsServiceImplServiceSoapBinding" type="tns:ItemBizService">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="addOrderInfo">
            <soap:operation soapAction="" style="document"/>
            <wsdl:input name="addOrderInfo">
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="addOrderInfoResponse">
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="ItemWsServiceImplService">
        <wsdl:port binding="tns:ItemWsServiceImplServiceSoapBinding" name="ItemWsServiceImp
lPort">
            <soap:address location="http://service.ken.com:8080/ws/ItemService"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

请求后端服务

代码的 `service.ken.com` 是测试用的后端RESTful应用。

```
curl --location --request POST 'http://service.ken.com:8080/ws/ItemService' \
--header 'Content-Type: application/xml' \
--data-raw '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xml
ns:ser="http://service.ken.com">
   <soapenv:Header/>
   <soapenv:Body>
      <ser:addOrderInfo>
         <orderInfo>
            <orders>
               <order>
                  <custNo>119640</custNo>
                  <custName>doctest</custName>
                  <items>
                     <item>
                        <goodsNo>1</goodsNo>
                        <goodsName>cookies</goodsName>
                     </item>
                     <item>
                        <goodsNo>2</goodsNo>
                        <goodsName>apple</goodsName>
                     </item>
                  </items>
               </order>
               <order>
                  <custNo>87981</custNo>
                  <custName>apptest</custName>
                  <items>
                     <item>
                        <goodsNo>3</goodsNo>
                        <goodsName>banana</goodsName>
                     </item>
                     <item>
                        <goodsNo>4</goodsNo>
                        <goodsName>coffee</goodsName>
                     </item>
                  </items>
               </order>
            </orders>
         </orderInfo>
      </ser:addOrderInfo>
   </soapenv:Body>
</soapenv:Envelope>'
```

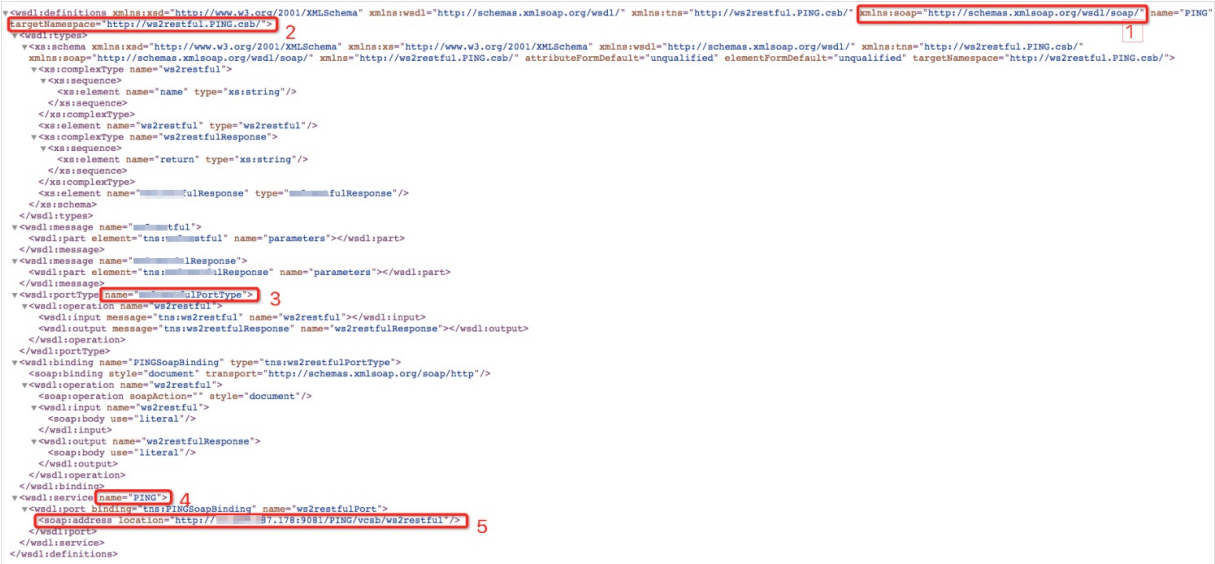
后端服务返回

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:addOrderInfoResponse xmlns:ns2="http://service.ken.com">
            <return>
                <orders>
                    <order>
                        <custNo>119640</custNo>
                        <custName>doctest</custName>
                        <items>
                            <item>
                                <goodsNo>1</goodsNo>
                                <goodsName>cookies</goodsName>
                            </item>
                            <item>
                                <goodsNo>2</goodsNo>
                                <goodsName>apple</goodsName>
                            </item>
                        </items>
                    </order>
                    <order>
                        <custNo>87981</custNo>
                        <custName>apptest</custName>
                        <items>
                            <item>
                                <goodsNo>3</goodsNo>
                                <goodsName>banana</goodsName>
                            </item>
                            <item>
                                <goodsNo>4</goodsNo>
                                <goodsName>coffee</goodsName>
                            </item>
                        </items>
                    </order>
                </orders>
            </return>
        </ns2:addOrderInfoResponse>
    </soap:Body>
</soap:Envelope>
```

JSON

> **说明** 接入WebService开放RESTful不支持JSON场景。

WebService请求说明



图注	取值	描述
1	-soap12	SOAP的版本，根据WSDL设置命名空间：`xmlns:soap12="http://www.w3.org/2003/05/soap-enveloap"` 或 `xmlns:soap="http://schemas.xmlsoap.org/soap/envloap/"`
2	-ns	目标命名空间。
3	-pname	端口号。
4	-sname	服务名称。
5	-ea	默认的Endpoint地址，调用时可以根据实际的服务地址进行设置。

3.2.3.5. 接入RESTful开放WebService

本文以实际场景为例介绍接入RESTful协议开放WebService协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入RESTful协议开放WebService协议的服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。
  - curl

```
curl -H 'Content-type:application/xml' -X POST -d "${Body}" \
http://CSB服务地址:9081/${服务名称}/${服务版本}/ws2ws?wsdl
```

- WS-SDK

```
java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} \
-ea ${ea} -ns ${ns} \
-sname ${sname} -pname ${pname} \
-rd "${Body}"
```

- 非公开访问（需要订购）的服务。

  在WS-SDK的基础上增加 `-ak ${访问凭证ak} -sk ${访问凭证sk}` 。

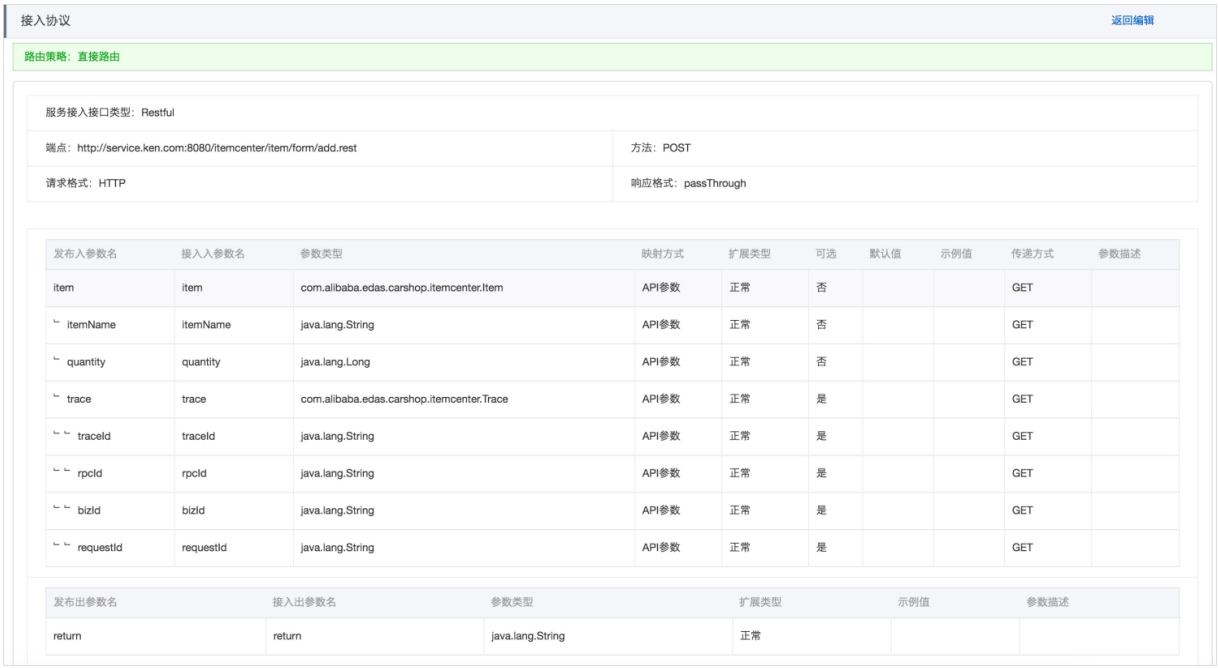
```
java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} \
-wa ${wa} -ea ${ea} -ns ${ns} \
-sname ${sname} -pname ${pname} \
-rd "${Body}"
```

FORM

> **说明** FORM场景，接入RESTful协议支持开放为WebService协议。

服务配置

接入RESTful开放WebService，不支持配置出参。



接入协议 返回编辑

路由策略：直接路由

服务接入接口类型：Restful

端点：http://service.ken.com:8080/itemcenter/item/form/add.rest　　方法：POST

请求格式：HTTP　　响应格式：passThrough

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
item	item	com.alibaba.edas.carshop.itemcenter.Item	API参数	正常	否			GET	
┗ itemName	itemName	java.lang.String	API参数	正常	否			GET	
┗ quantity	quantity	java.lang.Long	API参数	正常	否			GET	
┗ trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	API参数	正常	是			GET	
┗┗ traceId	traceId	java.lang.String	API参数	正常	是			GET	
┗┗ rpcId	rpcId	java.lang.String	API参数	正常	是			GET	
┗┗ bizId	bizId	java.lang.String	API参数	正常	是			GET	
┗┗ requestId	requestId	java.lang.String	API参数	正常	是			GET	

发布出参数名	接入出参数名	参数类型	扩展类型	示例值	参数描述
return	return	java.lang.String	正常		

接入后端服务

```
@PostMapping("/item/form/add")
public ResultDTO<Item> add(@JsonParam(value = "item") Item item);
```

请求代码

```
#使用WS SDK请求CSB服务
java -jar ws-client.jar -d -api item.form.add -version 1.0.0 \
-ea http://csb.target.server:9081/item.form.add/1.0.0/ws2restful \
-ns http://ws2restful.item.form.add.csb/ \
-sname item.form.add \
-pname ws2restfulPort \
-rd '<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wst
="http://ws2restful.item.form.add.csb/">
    <soapenv:Header/>
    <soapenv:Body>
      <wst:ws2restful>
        <item>
          <itemName>bmw</itemName>
          <quantity>10</quantity>
        </item>
      </wst:ws2restful>
    </soapenv:Body>
  </soapenv:Envelope>'
```

请求结果

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <soap:Body>
        <ns1:ws2restfulResponse
            xmlns:ns1="http://ws2restful.item.form.add.csb/">
            <return>
                {
                    "code": "0",
                    "msg": "SUCCESS",
                    "innerMsg": null,
                    "result": {
                        "trace": {
                            "traceId": "0ba783c115673287819481003****",
                            "rpcId": "0",
                            "requestId": "0ba783c115673287816961001d****"
                        },
                        "itemName": "bmw",
                        "quantity": 10
                    }
                }
            </return>
        </ns1:ws2restfulResponse>
    </soap:Body>
</soap:Envelope>
```

JSON

> **说明** 接入RESTful发布WebService，不支持JSON场景。

3.2.3.6. 接入HSF开放WebService

本文以实际场景为例介绍接入HSF协议开放WebService协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入HSF协议开放WebService服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。
  - curl

```
curl -H 'Content-type:application/xml' -X POST -d "${Body}" \
http://CSB服务地址:9081/${服务名称}/${服务版本}/${HSF方法}?wsdl
```

- WS-SDK

```
java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} \
-ea ${ea} -ns ${ns} \
-sname ${sname} -pname ${pname} \
-rd "${Body}"
```

- 非公开访问（需要订购）的服务。

  在WS-SDK的基础上增加 `-ak ${访问凭证ak} -sk ${访问凭证sk}` 。

```
java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk}\
-wa ${wa} -ea ${ea} -ns ${ns} \
-sname ${sname} -pname ${pname} \
-rd "${Body}"
```

接入HSF开放WebService的场景说明

服务配置

服务接入接口类型：HSF

接口全名：com.alibaba.edas.carshop.itemcenter.ItemService	方法名称：buy
版本号：1.0.0	服务分组名称：HSF

发布入参数名	接入入参数名	参数类型	映射方式	扩展类型	可选	默认值	示例值	传递方式	参数描述
item	item	com.alibaba.edas.carshop.itemcenter.Item	API参数	正常	是			GET	
└ itemName	itemName	java.lang.String	API参数	正常	是				
└ quantity	quantity	java.lang.Long	API参数	正常	是				
customer	customer	com.alibaba.edas.carshop.itemcenter.Customer	API参数	正常	是			GET	
└ name	name	java.lang.String	API参数	正常	是				
└ age	age	int	API参数	正常	是				

发布出参数名	接入出参数名	参数类型	扩展类型	示例值	参数描述
return	return	com.alibaba.edas.carshop.itemcenter.ResultDTO	正常		
└ code	code	java.lang.String	正常		
└ msg	msg	java.lang.String	正常		
└ result	result	com.alibaba.edas.carshop.itemcenter.Item	正常		
└ └ itemName	itemName	java.lang.String	正常		
└ └ quantity	quantity	java.lang.Long	正常		
└ └ trace	trace	com.alibaba.edas.carshop.itemcenter.Trace	正常		
└ └ └ traceId	traceId	java.lang.String	正常		
└ └ └ rpcId	rpcId	java.lang.String	正常		
└ └ └ bizId	bizId	java.lang.String	正常		
└ └ └ requestId	requestId	java.lang.String	正常		

接入后端服务

```
ResultDTO<Item> buy(Item item, Customer customer);
```

WSDL

```
http://CSB服务地址:9081/${服务名称}/${服务版本}/${HSF方法}?wsdl
```

调用服务

```
#使用WS SDK请求CSB服务。
java -jar ws-client.jar -d -api item.hsf.buy -version 1.0.0 \
-ea http://csb.target.server:9081/item.hsf.buy/1.0.0/buy \
-ns http://itemcenter.carshop.edas.alibaba.com/ \
-sname item.hsf.buy \
-pname addPortType \
-rd '<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wst
="http://itemcenter.carshop.edas.alibaba.com/">
  <soapenv:Header/>
  <soapenv:Body>
    <wst:buy>
      <item>
        <itemName>benz</itemName>
        <quantity>1</quantity>
      </item>
      <customer>
        <name>jack</name>
        <age>30</age>
      </customer>
    </wst:buy>
  </soapenv:Body></soapenv:Envelope>'
```

请求结果

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP-ENV:Header/>
    <soap:Body>
        <ns1:buyResponse
            xmlns:ns1="http://itemcenter.carshop.edas.alibaba.com/">
            <return>
                <code>0</code>
                <msg>SUCCESS</msg>
                <result>
                    <itemName>benz</itemName>
                    <quantity>2</quantity>
                    <trace>
                        <traceId>0ba783c115673284841491003d****</traceId>
                        <rpcId>0.1</rpcId>
                        <requestId>0ba783c115673284839321001d****</requestId>
                    </trace>
                </result>
            </return>
        </ns1:buyResponse>
    </soap:Body>
</soap:Envelope>
```

3.2.3.7. 接入WebService开放WebService

本文以实际场景为例介绍接入WebService协议开放WebService协议，以帮助您深入理解开放服务。

背景信息

本文仅介绍在实际场景中接入HSF协议开放RESTful服务时一些重要的配置步骤和参数，如需了解完整的流程和参数解释，请参见发布后端已有服务。

调用CSB开放服务时，支持多种请求方式，请根据实际场景选择：

- 公开访问（无需订购）的服务。
  - curl

    ```
    curl -H 'Content-type:application/xml' -X POST -d "${Body}" \
    http://CSB服务地址:9081/${服务名称}/${服务版本}/ws2ws?wsdl
    ```

  - WS-SDK

    ```
    java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} \
    -ea ${ea} -ns ${ns} \
    -sname ${sname} -pname ${pname} \
    -rd "${Body}"
    ```

    图片中各个选项值的定义：

    - 1、-soap12 SOAP的版本，根据WSDL里的如下命名空间：

      xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"

      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    - 2、-ns 目标命名空间。
    - 3、-pname 端口名。
    - 4、-sname 服务名。
    - 5、-ea 默认的endpoint地址，调用时可以根据实际的服务地址进行设置。

- 非公开访问（需要订购）的服务。

  在WS-SDK的基础上增加 `-ak ${访问凭证ak} -sk ${访问凭证sk}` 。

```
java -jar ws-client.jar -d -api ${服务名称} -version ${服务版本} -ak ${访问凭证ak} -sk ${访问凭证sk} \
-wa ${wa} -ea ${ea} -ns ${ns} \
-sname ${sname} -pname ${pname} \
-rd "${Body}"
```

接入WebService开放WebService的配置说明

服务配置

WSDL

```
http://CSB服务地址:9081/${服务名称}/${服务版本}/ws2ws?wsdl
```

接入后端服务

```
ResultDTO<Item> add(@WebParam(name = "item") Item item);
```

调用服务

您可以选择以下任一请求代码调用后端服务，代码中的 `service.ken.co` m是测试用的后端ws应用。

- 使用WS-SDK请求后端服务。

```
java -jar ws-client.jar -api item.ws -version 1.0.0 \
-ea http://csb.target.server:9081/item.ws/1.0.0/ws2ws \
-ns http://service.ken.com \
-sname ItemWsServiceImplService \
-pname ItemBizService \
-rd '<soapenv:Envelope  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:wst="http://service.ken.com">
  <soapenv:Header/>
  <soapenv:Body>
    <wst:add>
      <item>
        <itemName>benz</itemName>
        <quantity>1</quantity>
      </item>
    </wst:add>
  </soapenv:Body></soapenv:Envelope>'
```

- 通过curl工具请求调用服务，增加至Header参数里。

```
curl -H 'Content-type:application/xml' -X POST -d '<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:wst="http://service.ken.com">
    <soapenv:Header/>
    <soapenv:Body>
        <wst:add>
            <item>
                <itemName>benz</itemName>
                <quantity>10</quantity>
            </item>
        </wst:add>
    </soapenv:Body>
</soapenv:Envelope>' http://csb.target.server:9081/item.ws/1.0.0/ws2ws?wsdl
```

请求结果

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
        <ns2:addResponse
            xmlns:ns2="http://service.ken.com">
            <return>
                <itemName>benz</itemName>
                <quantity>73</quantity>
                <trace>
                    <bizId>0ba783c115667270091151002d****</bizId>
                    <requestId>0ba783c115673315358431001d****</requestId>
                    <rpcId>0.1</rpcId>
                    <traceId>0ba783c115673315360741003****</traceId>
                </trace>
            </return>
        </ns2:addResponse>
    </soap:Body>
    <SOAP-ENV:Header/>
</soap:Envelope>
```

3.3. 管理服务组

服务组（API组）是业务上的原子粒度分组，每一个服务（API）都归属且仅归属一个服务组。在发布新的服务时，必须指定其所属的服务组。本文介绍如何管理服务组。

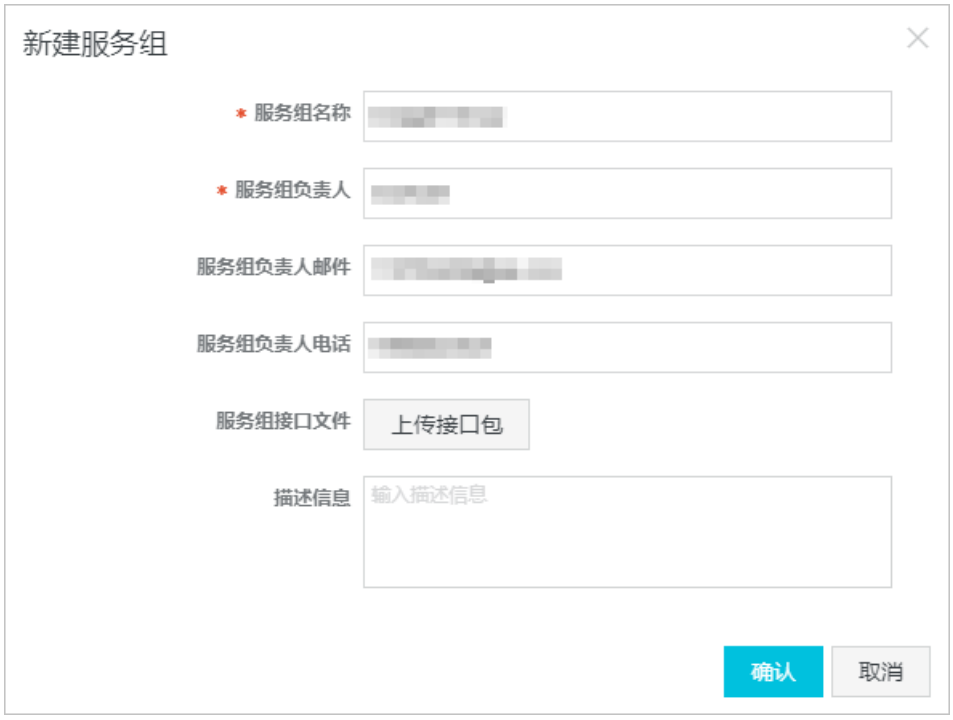
新建服务组

在CSB实例上发布服务前，需要先创建服务组。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击具体实例名称。

> 注意 如果您使用共享实例，请参见共享实例信息使用CSB指定的共享实例，否则会导致发布失败。共享实例仅用于体验试用，不建议正式生产使用。

5. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
6. 在**我的服务组**页面右上角单击**新建服务组**。
7. 在**新建服务组**对话框设置服务组参数，单击**确认**。



新建服务组参数说明。

参数	解释
服务组名称	自定义设置服务组名称。
服务组负责人	设置服务组的负责人。
服务组负责任人邮件	根据实际情况设置责任人的邮件和电话，非必配项。
服务组负责人电话	
服务组接口文件	为这个服务组上传一个JAR类型的接口文件，用于自动生成参数。

新建服务组后，返回**我的服务组**，该服务组的**状态**为**启动**。

查看服务组

1. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
2. 在**我的服务组**页面查看服务组的负责人信息、**状态**和该服务组发布的**服务数量**。

   如果服务组较多，可以通过**服务组名**的关键字进行搜索。

编辑服务组

1. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
2. 在**我的服务组**页面目标服务组的**操作**列单击**编辑**。
3. 在**编辑服务组**对话框修改服务组参数，单击**确认**。

停止服务组

> **注意** 停止该服务组后，所有属于这个服务组的服务都会停止。

1. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
2. 在**我的服务组**页面目标服务组的**操作**列单击**停止**。
3. 在**操作确认**对话框单击**确认**。
   停止服务组后，返回**我的服务组**，该服务组的**状态**为**停止**。

启动服务组

由于某些原因停止服务组后，您可以再次启动服务组，以便启动该服务中发布的所有服务。

1. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
2. 在**我的服务组**页面目标服务组的**操作**列单击**启动**。
3. 在**操作确认**对话框单击**确认**。
   停止服务组后，返回**我的服务组**，该服务组的**状态**为**启动**。

删除服务组

当不在需要使用某个服务组后，可以删除该服务组。

> **注意**
> - 如果该服务组下有服务，无法删除，会弹出错误提示。删除服务组前，需要先删除其下所有服务。
> - 服务组删除后，将不可恢复。所以在删除前，务必确认是否不再需要该服务组。

1. 在**实例概览**页面左侧导航栏中选择**发布者 > 我的服务组**。
2. 在**我的服务组**页面目标服务组的**操作**列单击**删除**。
3. 在**操作确认**对话框单击**确认**。

3.4. 管理服务

您在CSB实例中发布服务后，可以管理发布的服务，包括管理服务的生命周期、设置黑白名单、复制服务、导入服务、导出服务和转换服务的定制化参数等。

管理生命周期

在**我的服务**页面可以管理服务的生命周期。

生命周期包含以下状态：

- **激活（启动）**
- **停止**

- **待审批（等待一次审批）**
- **审批驳回（一级审批驳回）**
- **等待二级审批**
- **二级审批驳回**
- **注销**

管理服务的生命周期包括**启动**、**停止**和**注销**。

- 服务可以在**停止**和**启动**两种状态间切换，但是一旦**注销**则不可以再启动。
- 服务注销是服务生命周期的最终状态，意味着不可以对该服务进行变更，启动、停止和订购操作。

设置黑白名单

您可以设置服务访问的黑白名单。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**单击具体实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 我的服务**。
6. 在**我的服务**页面选择目标服务，在该服务的**操作**列中选择**更多 > 黑白名单**。
7. 在**编辑黑白名单**对话框中单击**黑名单**、**白名单**或**配置**页签，进行配置，然后单击**确定**。
   - **黑名单**：输入拒绝访问的IP地址，单击**添加**。
   - **白名单**：输入允许访问的IP地址，单击**添加**。
   - **配置**：在**默认策略**右侧单击**通过**或**拒绝**，设置默认策略，然后单击**确定**。

     对于既不在白名单里，也不在黑名单里的IP地址，**通过**表示无需鉴权即可访问，**拒绝**表示该服务默认不允许任何IP地址访问。

     IP地址的添加规则说明：

     可以添加某个具体IP地址，例如*192.168.1.1*；也可以设置带有掩码的IP地址段，例如*192.168.1.1/24*（`/`后面的取值范围是1~32，代表子网掩码的位数，24表示掩码为：255.255.255.0）。

     > **说明**
     > - 在白名单中添加某个调用者的IP后，该调用者不做鉴权即可访问该服务。
     > - 在黑名单中添加某个调用者的IP后，该调用者不能访问该服务。
     > - 如果某个调用者的IP既被添加到白名单，又在黑名单里，则黑名单优先，即拒绝访问该服务。

复制服务

复制服务是用来快速的将一个已经存在的服务复制成一个新的服务，它的接入和开放协议及参数等信息保持不变，改变的是服务名、服务版本或者服务组。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。

3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**单击具体实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 我的服务**。
6. 在**我的服务**页面选择目标服务，在该服务的**操作**列中选择**更多 > 复制**。
7. 在弹出的**操作确认**对话框，单击**确认**。

> **说明** 复制服务的初始状态为**停止**状态。

8. 在**复制服务**对话框输入**服务全名**和**服务版本**，并选择**所属服务组**，然后单击**确认**。

> **说明**
> - 服务复制仅限于在当前实例中。
> - 复制过程中要保证新服务的**服务全名+服务版本**在本实例中保持唯一，并且其它同名的服务已经处于注销状态。

导出和导入服务

您可以导出和导入服务，且都支持批量操作。

- **导出服务**
  i. 在**我的服务**页面搜索需要导出的任务。

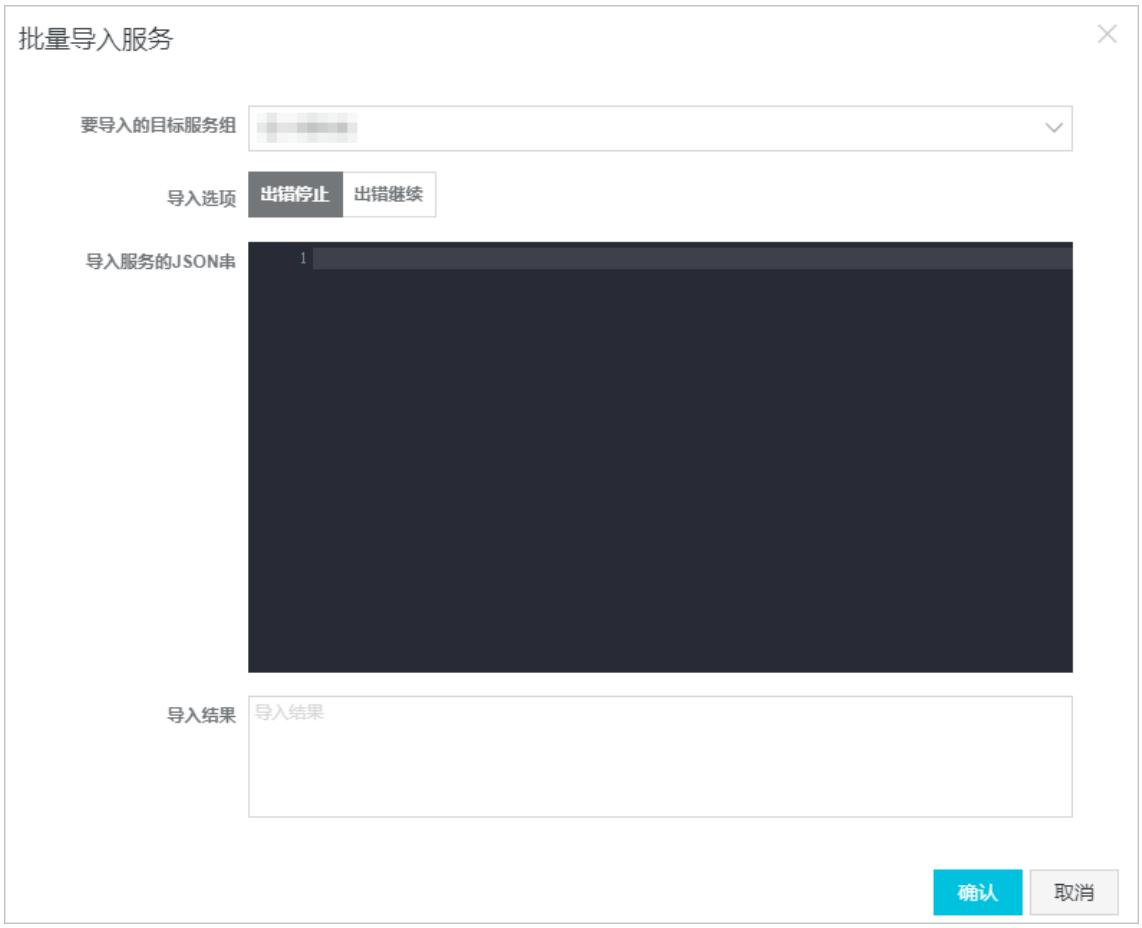
     支持按照**服务名**、**服务组名**、**服务别名**和**后端接入端点**进行搜索服务。
  ii. 在**我的服务**页面单击**导出服务**。
  iii. 在弹出的**操作确认**对话框中单击**确认**。
  iv. 在**导出服务**对话框中复制当前服务的JSON格式内容，保存到文本编辑器，将当前服务导出。

     如果导出多个服务的JSON代码，需要按照 `[json1, json2,...]` 格式，其中，json1,json2为服务导出时的单个服务的完整的JSON串。

     > **说明** 格式串中最外层使用方括号（[]）形成一个导入的文本。
- **导入服务**

i. 在**我的服务**页面单击**导入服务**。

ii. 在**批量导入服务**对话框中选择**要导入的目标服务组**，设置**导入选项**，并将导出服务的JSON代码串粘贴到**导入服务的JSON串**区域，然后单击**确认**。



导入过程中如果一条服务导入失败，可以选择继续导入其他服务或者停止导入。

> **说明**
>
> - 如果要导入的服务已经在本服务组中存在，可以对导入JSON文本中的service_name值进行修改，然后保存进行导入。
> - 导入服务时不会导入级联服务，只会把需要导入的服务导入到当前实例下指定的服务组中。

转换服务的定制化参数

CSB支持服务入参和出参的定制化参数转化，目前仅支持HTTP协议开放成HTTP协议的场景，数据类型仅支持form和JSON，脚本只支持Groovy。

1. 在**我的服务**页面选择目标服务，在该服务的**操作**列中选择**更多 > 脚本**。
2. 在**参数映射脚本配置**对话框选择**请求脚本**或**响应脚本**，在**请求脚本**或**响应脚本**页签中输入**编程语言**（默认为GROOVY），在**编辑代码**区域编写脚本，然后单击**确认**。

参数映射脚本配置说明：

- 在**请求脚本**中，数据在简单映射后执行，对于form数据会增量覆盖简单映射后的数据；对于JSON数据会全量覆盖调用方传递过来的数据，默认为 `importRequest=exportRequest` 。
- 在**响应脚本**中，目前仅支持JSON，数据会全量覆盖源服务的响应数据，默认为 `exportResponse=importResponse` 。
- 脚本映射的结果会作为界面映射的输入，建议使用自定义映射就不要使用界面映射，逻辑更清晰。
- 支持4个变量：

```
exportRequest //开放服务请求参数
importRequest //接入服务请求参数
exportResponse //开放服务返回参数
importResponse //接入服务返回参数
```

- HTTP请求结构详解：

```
{
header:{"":"","":""},//key和value都必须是字符串。
query:{"name":["jackson","catalina"],"age":"1"},  //key字符串,value可以是字符串（单值），或者是字符串的list（多值）。
body:? // form:{"name":["jackson","catalina"],"age":"1"},json:["",""] or {...}
}
```

3.5. 审批服务发布

您可以为创建的实例设置服务发布审批策略，如果设置了需要一级审批或二级审批，则在该实例上发布服务需要您进行审批。

背景信息

您可以为创建的实例设置不同的服务发布审批策略，请参见设置服务发布审批策略。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。

3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**单击具体实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 服务发布审批**。
6. 在**服务发布审批**页面，根据实际情况审批服务发布申请。
   - 在待审批的服务发布申请的**操作**列单击**通过**，在弹出的对话框中填写审批意见（可选），然后单击**确认**，批准服务发布的申请。
   - 在待审批的服务发布申请的**操作**列单击**拒绝**，在弹出的对话框中填写审批意见（可选），然后单击**确认**，拒绝服务发布的申请。

如果服务发布申请较多，您也可以设置**过滤条件**和**审批服务**。

- **过滤条件**：包含**显示注销**和**不显示注销**，即是否显示已注销的服务，默认为不显示注销。
- **审批服务**：包括**已经审批过的**、**待一级审批**、**待二级审批**，默认为**待一级审批**。

  单击**已经审批过的**，即可查看审批记录。

如果服务发布申请较多，可以通过**服务名**、**服务组名**或**服务别名**的关键字进行搜索。

3.6. 审批服务订阅

服务发布者需要对订阅者提交的服务订阅申请进行审批，审批通过后，调用者才可以使用该服务。

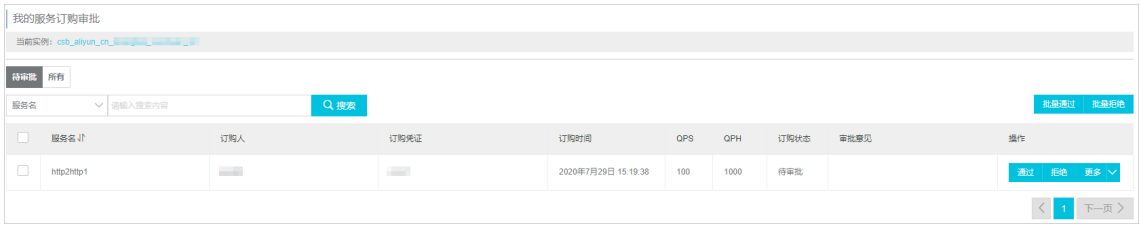
背景信息

发布服务时，可设置**访问限制**。

- 如果打开**公开访问**，即允许不授权访问。使用者可以直接使用任意合法的AccessKey ID和AccessKey Secret访问服务。
- 如果关闭**公开访问**，即必须经过授权才能访问，使用者必须先订阅（提交申请），所有者审批通过后才能正常调用。

操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**单击具体实例名称。
5. 在**实例概览**页面左侧导航栏选择**发布者 > 订购审批**。
6. 在**我的服务订购审批**页面单击**待审批**页签。
7. 在**待审批**页签根据实际情况，审批订购服务申请。
   - 单击**通过**，在弹出的对话框中填写意见（可选），然后单击**确认**，批准服务订阅的申请。
   - 单击**拒绝**，在弹出的对话框中填写意见（可选），然后单击**确认**，拒绝服务订阅的申请。



如果订购申请比较多，可以通过**服务名**、**服务组名**、**服务别名**或**凭证名**的关键字进行搜索。

> **说明** 在该页面您还可以修改订阅服务的QPS和QPH值：
>
> - 单个修改SLA
>   a. 在目标订阅服务的**操作**列，选择**更多 > 修改SLA**。
>   b. 在**修改服务订购**对话框，设置QPS和QPH值，然后单击**确认**。
> - 批量修改SLA
>   a. 选中多个订购服务。
>   b. 单击**批量修改SLA**。
>   c.

3.7. 约束规范

服务发布规范定义了服务发布中的命名规则、协议选择和映射关系，以及其他的一些规范信息。

命名规范

- **服务组名称**：只允许使用中英文字符、数字、短划线（-）和下划线（_），不允许有空格。
- **服务全名**：不可以为中文，只能使用数字，字母和短划线（-）。
- **服务版本**：不可以为中文，只能使用数字，字母和英文句号（.）。
- **服务发布的参数名**：只能使用数字，字母和下划线（_）。

协议转换种类

CSB支持常用协议服务的接入和开放（RESTful/SpringCloud/HSF/WebService/Dubbo/JDBC），可扩展支持定制化的协议转换。

- 服务接入：在CSB上注册某个服务并且提供足够的信息让CSB可以访问这个已有的服务。
- 服务开放：把一个已接入的服务在某个CSB实例上提供对应不同协议的API调用入口。

默认支持的服务接入、开放协议如下表所示：

支持的接入协议类型	对应支持的协议开放类型
RESTful	RESTful、WebService
SpringCloud	RESTful
HSF	RESTful、WebService、HSF级联
WebService	RESTful、WebService级联

支持的接入协议类型	对应支持的协议开放类型
Dubbo	RESTful、WebService
JDBC	RESTful

- CSB支持数组、列表、集合类型的服务参数，也可以定义复杂的多级参数结构，其它特别的支持情况，请参见CSB接入规范和限制。
- 在协议转换外，CSB还支持接入接口和开放接口的参数映射、是否可选、缺省值以及在开放接口上是否可见等设置。还将支持定制开发的参数映射机制（Groovy脚本）。
- CSB暂不支持OAuth 2.0开放标准。

CSB接入规范和限制

开放 > 接入协议对	接入规范和限制明细

RESTful> RESTful	• 支持GET/PUT/POST/DELETE
	• 参数传递
	  ○ Query：Key-Value形式
	  ○ Body：Key-Value/JSON/byte[]形式
	• 支持RESTful风格的接入，如 `http://localhost:9090/aaa/{userId}/test` 。
	• 支持HTTP返回结果透传，不做任何变换加工。

RESTful> Dubbo	• 支持特别指定注册中心或使用默认注册中心。 • 支持开放一个接口的某个指定方法。 • 支持Dubbo和Hessian序列化协议。 • 支持简单数据类型泛型的HashMap。 • 返回结果固定为JSON格式。

RESTful> WebService	• 支持两种格式的用户名密码接入SOAP Header。
	  ○ Authorization类型
	  ○ WSSE类型
	• 不支持WSS和Policy等相关的安全和加密处理。
	• 对于*anytype*参数类型，参数定义时要经过特殊定义为ws:anyType或者java.lang.Object。
	• 接入后端WebService服务方法必须是同步的，不支持异步。
	• 支持的bindType为 `Document-Literal (Wrapper or Unwrapper)` 。
	• 返回结果是将SOAP XML转化为JSON格式。

RESTful> HSF	支持开放一个接口某个指定方法。 • 支持java.util.Map参数。 • 返回结果固定为JSON格式。

开放 > 接入协议对	接入规范和限制明细
WebService > WebService	• 支持MTOM等附件数据传输方式的调用。 • 支持透传式调用后端接入服务。 • 需要使用CSB提供的WebService Client SDK（目前有Java版本）传递AccessKey和signature等相关Header信息。
WebService > Restful/Dubbo/HSF	• 相应服务会以SOAP 1.1的格式发布成一个WebService并提供标准的WSDL。 • 需要使用CSB提供的WebService Client SDK（目前有Java版本）传递AccessKey和signature等相关Header信息。 • 暂不支持入参的可选和默认值设置。

服务控制

- 支持具体用户消费凭证到具体服务的访问限流设置。
- 支持实例级（即服务器级）的总体访问流量保护。
- 支持实例级（即服务器级）和服务级的基于IP的黑白名单设置。

服务发布

提供版本管理和基本的服务（API）组织管理，支持跨CSB实例联动发布。

- API服务发布的接入和开放协议，缺省支持的情况，请参见协议转换种类。
- 支持API服务分组，用以实现对所发布服务的基本组织分类。
- 支持API服务多版本，可指定变更当前的激活版本。
- 支持API服务的启用、停用、注销、定义变更等基本生命周期管理。
- 支持API服务基于IP的黑白名单设置。
- 支持指定API服务每秒最大调用量。
- 支持指定无需授权审批，申请即可自动授权的自由调用模式。
- 支持基于路由选择的服务发布，可以根据接入参数的取值不同决定接入的实际协议或地址。
- 支持API服务的级联发布，即可以在一个CSB实例上接入服务，可能途经其它中间CSB实例，然后在另一个目标CSB实例上开放服务，消费方应用直接访问目标CSB实例即可。
- 支持对已发布API服务的级联扩展，例如在A实例接入B实例开放的API，可以再次指定在实例C上开放，服务级联链路按规则自动适配。

CSB发布的服务的端口说明

1. CSB发布出的RESTful服务的地址访问格式：`http://CSB服务地址:8086/CSB`。
   - *CSB服务地址*即创建该实例时绑定的SLB的地址。
   - 默认的访问端口为*8086*。
   - 请求的context-path可以任意指定，默认使用*CSB*。
2. 根据CSB的设计约定，当CSB开放成WebService服务时，对应的WSDL的地址为以下格式：

- 如果接入是HTTP协议，则开放出的WSDL地址是 `http://CSB服务地址:9081/$api_name/$api_version/ws2restful?wsdl` 。
  - *CSB服务地址*即创建该实例时绑定的SLB的地址。
  - $api_name为发布的服务名。
  - $api_version为发布的服务版本。
  - ws2restful为固定值。
- 如果接入是HSF协议， 则开放出来的WSDL地址是 `http://CSB服务地址:9081/$api_name/$api_version/$method?wsdl` 。
  - *CSB服务地址*即创建该实例时绑定的SLB的地址。
  - $api_name为发布的服务名。
  - $api_version为发布的服务版本。
  - $method为发布服务时对应的接入方法名。
- 如果接入是WS协议（即WS透传）， 则开放出来的WSDL地址是 `http://CSB服务地址:9081/$api_name/$api_version/ws2ws?wsdl` 。
  - *CSB服务地址*即创建该实例时绑定的SLB的地址。
  - $api_name为发布的服务名。
  - $api_version为发布的服务版本。
  - ws2ws为固定值。

CSB服务的调用返回约定

调用由CSB发布的HTTP服务时，如果发生调用错误，则默认返回值是一个如下的JSON格式：

```
{
 "code":500,
 "message":"[503]service not registed, key is : _api_PING_null_null_vcsb",
 "requestId":"1e1eb5d215059677158731001d3d47"
 }
```

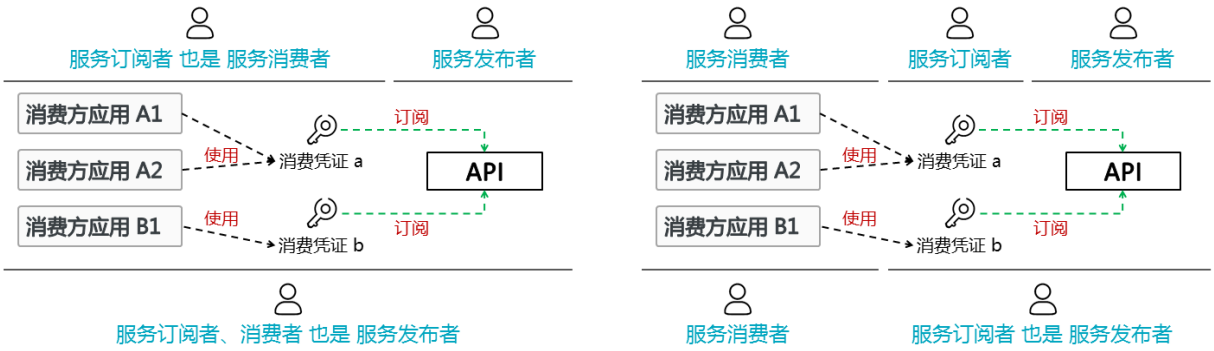
4.服务订阅

4.1. 服务订阅场景示例

当您订购服务时，会有不同的需求和场景，例如服务订阅方就是实际的服务消费方、服务订阅方不是实际的服务消费方。

简介

关于服务的发布、审核、授权的基本模式请参见服务。其中服务的发布、订阅和消费，有下图所示的几种典型方式。



服务订阅方是实际的服务消费方

如图左侧部分所示，您使用创建的消费凭证订阅服务API后，您的应用也使用这些凭证来发起调用。这是最常见的使用场景。

一种特别的场景是：服务的提供方其实也是自己，这通常会发生在您有多个环境的情况下，例如您在阿里云上发布了应用中的一些服务供其它环境（如某个数据中心）里的应用调用。

服务订阅方不是实际的服务消费方

如图右侧部分所示，您用自己的消费凭证订阅服务API后，把这些凭证交给其他人使用，这些使用消费凭证的人不需要是阿里云用户，只需在自己的应用程序里使用这些消费凭证来发起调用即可。CSB允许采用这种方式，需要您管理好消费凭证的分发，避免安全风险。

一个常见的典型场景是：您订阅的服务是您自己发布的，只不过这些服务是提供给其他人（例如您的客户）调用的。您发布服务，创建多个消费凭证进行服务订阅授权，然后交给您的客户调用。

4.2. 管理凭证

API消费者使用凭证来订购服务API，API消费方应用需要使用API消费凭证（简称凭证）来调用CSB上开放的服务API。本文介绍如何创建和管理凭证。

背景信息

凭证通常为一对AccessKey ID和AccessKey Secret，在API调用时用来做签名信息计算，CSB接收到API调用请求时对签名信息做验证。

目前还支持JWT类型的凭证，提供JWT Token信息，可直接用于服务调用。更多信息，请参见如何使用JWT Token调用服务。

您可以创建多个凭证代表不同的应用或者应用组。

创建凭证

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击要调用的服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的凭证**。
6. 在**我的凭证**页面单击**创建凭证**。
7. 在**创建凭证**页面选择**凭证类型**，输入**凭证名称**和**（可选）归属**，然后单击**确认**。
   - **凭证名称**：通常使用该凭证要代表的应用名、应用组名或系统名等。仅允许英文字母、下划线（_）和数字，长度为5~64个字符。
   - **凭证类型**：根据实际需要选择创建凭证的类型，包含**Dauth**、**Dauth导入**、和**JWT**类型。
     - **Dauth**：新建订购服务的消费凭证。系统会自动生成**AccessKey ID**和**AccessKey Secret**信息，系统是异步生成**AccessKey Secret**，会有短暂延时。
     - **Dauth导入**：使用原先已有且当前未使用的**AccessKey ID**、**AccessKey Secret**来新建订购服务的消费凭证。您需要正确设置**AccessKey ID**和**AccessKey Secret**。
     - **JWT**：新建支持调用服务的JWT类型的消费凭证。该类型凭证还需设置**有效期**，支持**长期**和**短期**；系统会自动生成**AccessKey ID**、**AccessKey Secret**和**JWT**信息，系统是异步生成**AccessKey Secret**，会有短暂延时。
   - **（可选）归属**：长度为5~64个字符，无字符类型限制。

   创建完成后，在**我的凭证**页面显示该凭证的信息，包含凭证的AccessKey ID（AK）和AccessKey Secret（SK）。

查看凭证

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击目标服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的凭证**。
6. 在**我的凭证**查看所有的凭证及凭证信息。

变更凭证

当出于安全考虑，例如怀疑当前凭证已经泄露，可以变更凭证。

> **说明** 由于使用该凭证的应用也需要一段时间才能更新成新的凭证信息，通常需要一个时间窗口让新、旧凭证同时生效。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击目标服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的凭证**。
6. 在**我的凭证**页面选择目标凭证，然后在该凭证的**凭证操作**列单击**更新**，在弹出的**更新实例凭证**对话框单击**确认**，生成新的凭证。

   生成新的凭证后，新、旧凭证都是可用的。
7. 确认旧的凭证不再使用后，在**我的凭证**页面选择目标凭证，然后在该凭证的**凭证操作**列单击**取代**，废止旧的凭证。
   旧的凭证废止后，新的凭证会取代成为当前凭证。

删除凭证

您使用凭证订购服务，删除凭证后，服务的订购将失效。所以，请确认不再需要通过该凭证订购服务后，再删除该凭证。

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击目标服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的凭证**。
6. 在**我的凭证**页面选择目标凭证，然后在该凭证的**凭证操作**列单击**删除**。
7. 在**删除凭证**对话框单击**确认**。

4.3. 订购服务

您可以在CSB实例中订购需要使用的服务。

前提条件

您已经创建了订购该服务的凭证，请参见创建凭证。

背景信息

服务发布时，在**限制访问**页面可以选择打开或关闭**公开访问**开关。针对不同的设置，订购该服务的策略会有所不同。

- 如果打开**公开访问**开关，则服务消费端无需订购，根据服务的访问地址和参数即可使用标准的RESTful方式发起服务调用。
- 如果关闭**公开访问**开关，则服务消费端需要订阅、审批后，使用创建的凭证的AccessKey ID和AccessKey Secret发起服务调用。

订购支持白名单，即仅允许白名单内的IP使用该凭证调用。

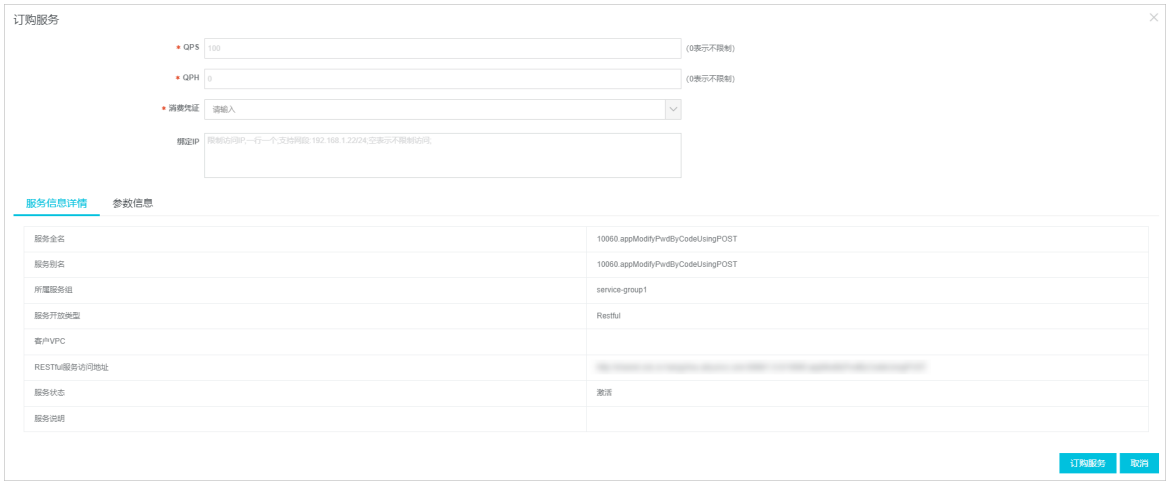
操作步骤

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击要调用的服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 订购服务**。
6. 在**订购服务**页面选择目标服务，在该服务**操作**列单击**查看详情与订购**。

   > **说明** 如果服务较多，可以通过**服务名**、**服务组名**或**服务别名**的关键字进行搜索。

7. 在**订购服务**对话框中设置参数，然后在页面下方单击**订购服务**。

   可以在对话框下方查看该服务的**服务信息详情**和**参数信息**。



   订购服务参数说明：

   - **QPS**用于设置该服务的访问频率（每秒钟访问量）。
   - **QPH**用于设置该服务的访问频率（每小时访问量）。
   - **消费凭证**用于订购服务，从下拉列表中选择目标凭证。
   - **绑定IP**用于限制访问该服务的IP地址，多个IP之间通过换行分隔。不设置表示不限制访问IP。

   订购服务场景说明：

   - 如果订阅者也是服务发布者，无需审批，系统自动通过。
   - 如果订阅者不是服务发布者，可以在左侧导航栏选择**订阅者 > 我的订购**，在**我的订购**页面查看所订

购的服务信息和订购信息。

> **说明** 其中**订购状态**为**待审批**，需要待发布者审批，通过订购申请后才能调用。

4.4. 管理订购的服务

订阅者可以查看和管理订购的服务。

查看订购的服务

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击要调用服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的订购**。
6. 在**我的订购**页面查看订购的服务及其信息。

如果订购的服务较多，可以单击**订购中的服务**或**全部订购的服务**进行筛选，也可以按**服务名称**、**服务组**、**凭证名称**或者AccessKey ID进行搜索。

您可单击所订购的服务名称或凭证名称，查看详情。

管理订购的服务

1. 登录CSB控制台。
2. 在顶部菜单栏选择地域。
3. 在左侧导航栏单击**实例列表**。
4. 在**实例列表**页面单击要调用服务所在的实例名称。
5. 在**实例概览**页面左侧导航栏选择**订阅者 > 我的订购**。
6. 在**我的订购**页面管理订购的服务。
   - 测试服务

     在需要在线测试的服务的**操作**列单击**在线测试**，进入**在线测试**页面。更多信息，请参见如何在线测试服务。
   - 退订服务

     在需要退订的服务的**操作**列单击**更多**右侧的下拉图标，在下拉列表中单击**退订**。
   - 修改绑定IP

     当订购的服务的IP地址或IP地址段改变之后，要修改绑定的IP，保证后续可以正常调用服务。

a. 在需要修改绑定IP的服务的**操作**列单击**更多**右侧的下拉图标，在下拉列表中单击**修改绑定IP**。

b. 在**修改绑定IP**对话框中输入订购的服务变更后的IP地址或IP地址段，多个IP地址需要换行分隔。

- 退订服务

在需要退订的服务的**操作**列单击**更多**右侧的下拉图标，在下拉列表中单击**退订**。

5.故障排除

5.1. 调用HSF服务异常

本文介绍在调用HSF服务时，出现异常 `HSFServiceAddressNotFoundException` 的原因和处理方法。

Condition

当您通过CSB调用EDAS HSF服务时，出现异常 `HSFServiceAddressNotFoundException` 。

Cause

- 环境问题，例如地址服务器、网络或命名空间问题。
- 服务问题，例如服务本身调用异常或在CSB中配置有误。

Remedy

操作步骤

1. 检查CSB实例和EDAS提供的HSF服务（简称HSF服务）使用的地址服务器（jmenv.tbsite.net）是否一致。

   有两种排查方式：

   - 在*/etc/hosts*文件中查看设置的地址服务器的IP是否与实际一致。
   - 在Java进程中查看参数-Daddress.server.domain配置的IP是否与实际一致。

   根据排查结果选择后续操作。

   - 如果一致，进行下一步。
   - 如果不一致，修改*/etc/hosts*或-Daddress.server.domain与实际地址服务器一致。
2. 通过Telnet，检查CSB实例和HSF服务间的网络连接是否正常。

   HSF服务的端口为Java进程打开的端口，范围为12200~12299。可以登录EDAS 控制台，在服务管理页面查看。

   - 如果正常，进行下一步。
   - 如果不正常，排查、修改网络连接。
3. 通过查看Java进程的-Dtenant.id参数，检查命名空间配置与实际是否一致。
   - 如果一致，进行下一步。
   - 如果不一致，修改命名空间配置。
4. 检查HSF服务在EDAS中是否能够正常调用。

   登录EDAS控制台，在**服务查询**页面查询要调用的HSF服务；或者如果知道要调用的应用，在该应用详情页左侧的导航栏中单击**服务列表**，查看**发布的服务**和**消费的服务**，确认该服务是否可以正常调用。

   - 如果能够正常调用，进行下一步。
   - 如果不能正常调用，请排查、修复服务本身的调用问题。
5. 在CSB控制台中检查HSF服务的配置是否正确。

   登录CSB控制台，检查HSF服务配置是否正确，请参见发布后端已有服务。

> **说明** 如果该HSF服务在CSB发布了多次，需要检查该HSF服务的配置（例如所属服务组）是否一致。如果不一致，也会导致调用失败。

- 如果配置正确，进行下一步。
- 如果配置不正确，修正配置。

6. 查看*/home/admin/configclient/snapshot/{interfaceName}:{version}/{groupName}-xxxxxxx.data*和*/home/admin/logs/hsf.log*日志中是否包含该HSF服务和报错信息。
   - 如果包含，根据日志信息排查、处理。
   - 如果未包含，请联系CSB技术支持人员。