云日志服务

用户指南

文档版本 01

发布日期 2024-07-17

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 授权 IAM 用户使用 LTS

如果您需要对您所拥有的LTS（Log Tank Service）进行精细的权限管理，您可以使用**统一身份认证服务**（Identity and Access Management，简称IAM），通过IAM，您可以：

- 根据企业的业务组织，在您的华为云账号中，给企业中不同职能部门的员工创建IAM用户，让员工拥有唯一安全凭证，并使用LTS资源。
- 根据企业用户的职能，设置不同的访问权限，以达到用户之间的权限隔离。
- 将LTS资源委托给更专业、高效的其他华为云账号或者云服务，这些账号或者云服务可以根据权限进行代运维。

如果华为云账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用LTS服务的其它功能。

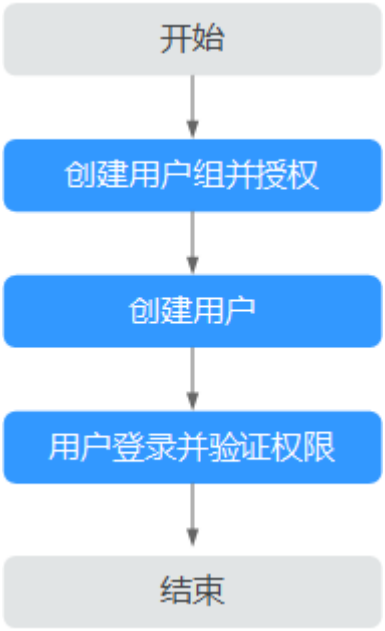
本章节为您介绍对用户授权的方法，操作流程如**图1-1**所示。

前提条件

给用户组授权之前，请您了解用户组可以添加的LTS权限，并结合实际需求进行选择LTS支持的系统权限，请参见：**权限管理**。

示例流程

图 1-1 给用户授权 LTS 权限流程



1. 登录统一身份认证服务IAM控制台。在IAM控制台创建用户组，并授予云日志服务操作权限“LTS FullAccess”。详细操作请参考**创建用户组并授权**

   说明

   选择“LTS FullAccess”，由于该策略有依赖，除了勾选LTS FullAccess外，还需要在同项目中勾选依赖的策略：Tenant Guest、以及“全局区域 对象存储服务项目”中勾选依赖的策略：Tenant Administrator。

2. 在IAM控制台创建用户，并将其加入**步骤1**中创建的用户组。详细操作请参考**创建用户并加入用户组**
3. 使用新创建的用户登录控制台，切换至授权区域，验证权限。详细操作请参考**用户登录**并验证权限。

2 购买 LTS 资源包

云日志服务支持购买资源包进行计费，在费用结算时，优先从资源包中抵扣用量。先购买资源包，后抵扣费用，适用于业务用量相对稳定的场景。更多计费信息**价格计算器**。

说明

目前此功能仅支持华北-北京四、华东-上海一、华南-广州、华北-北京一、华北-北京二、华东-上海二、西南-贵阳一、华北-乌兰察布一局点，其他局点需要提交工单申请使用。详细操作请参考**提交工单**。

使用规则

- 资源包到期后，按需资源不会自动关闭，将会以按需付费的方式继续使用。
- 超出额度部分的按需用量，将会按需收费。
- 资源包计费周期的起点是用户设置的生效时间，清零和恢复时间点是购买时长到期后当天的23:59:59。
- 资源包到期后，剩余资源不会结余，将会自动清零。更多信息请参考**资源包扣减规则**。

购买 LTS 资源包

步骤1 登录**云日志服务控制台**。

步骤2 在日志管理页面，单击“购买资源包”。

步骤3 在购买资源包页面，请参考**表2-1**设置参数。

说明

资源包中的冷存储包是白名单功能，同时需要开启智能冷存储，详细请参考**管理日志流**，如有需要请**提交工单**申请开通。

表 2-1 资源包参数

参数	说明
区域	云日志服务所在的区域，建议选择与业务应用系统相同的地域。

参数	说明
计算方式	支持两种计算方式： 1. 推荐规格： **说明** – 在LTS控制台的购买资源包页面，存储量的单位是GB/月，在费用中心控制台，存储量的单位是GB/小时。例如在LTS购买1个数量100GB的标准存储包（100GB/月），在费用中心的资源包页面存储总量有100GB*1个*24小时*30天=72000GB。 – 若只购买1个数量100GB的读写流量包，在费用中心的资源包页面总量显示100GB。 – 若只购买1个数量100GB的索引流量包，在费用中心的资源包页面总量显示100GB。 – 资源包类型：支持读写流量包、索引流量包、标准存储包、冷存储包（白名单功能） – 规格：默认支持100GB – 购买数量：支持设置1-3000 – 规格说明：规格*规格数量 2. 一键测算：按照每天新增日志量（GB/天或TB/天）、日志存储时长（天）、标准存储层数据保存时间(天)进行测算读写流量包、索引流量包、标准存储包、冷存储包的使用量。 读写流量包和索引流量包与每天新增日志量有关，标准存储包和和冷存储包与每天新增日志量、日志存储时长、标准存储层数据保存时间(天)有关。 例如选择每天新增日志量10GB/天，日志存储时长30天，标准存储层数据保存时间7天，云日志服务提供默认规格是100GB，一键测算一个月的结果如下：（若是选择TB/天，先将TB换算为GB）  – 读写流量包100GB*1，读写有5倍的压缩率，因此所需要的数量：（10GB*30天）/5=60GB，不足100GB的算1个。 – 索引流量包100GB*3，所需要的数量：（10GB*30天）/100GB=3个。 – 标准存储包100GB*1，所需要的数量：10GB*7天=70GB，不足100GB的算1个。

参数	说明
	– 冷存储包100GB * 3，冷存储的天数为30-7=23天，所需要的数量：（10GB*23天）/100GB=3个，不足100GB的算1个。
购买时长	选择资源包的使用时长，支持按月、按年购买。
生效时间	• 支付完成后立即生效。 • 指定生效时间：设置指定生效的具体时间，例如2023/07/11 10:30:00 **说明** 资源包生效时间非整点时，例如2023/07/11 10:30:00开始生效，开始抵扣时间为下一个整点2023/07/11 11:00:00。
定向使用	• 所有企业项目均可使用。 • 限定企业项目使用套餐包，选择特定企业项目。 **说明** 如果您没有开通企业管理服务，将无法看到企业项目选项。开通方法请参见**如何开通企业项目**。

步骤4 设置完成后，单击“加入清单”。

步骤5 确认清单信息无误后，单击“立即购买”。

步骤6 在购买详情页面，根据业务需要调整购买数量购买时长，单击“去支付”。

说明

单次订单中资源包的购买数量总和不超过5000。

步骤7 在购买云日志服务页面选择支付方式，单击“确认付款”，完成后，即可单击返回云日志服务控制台。

步骤8 在资源包卡片上方，单击“查看已购买的资源包”，进入资源包页面查看已购买资源包信息。详细操作请参考**资源包**。

----结束

3 日志管理

3.1 日志管理概述

云日志服务LTS是以日志组和日志流为基本单位进行日志管理。使用云日志服务LTS之前，请先创建日志组和日志流。创建日志组后，可以在该日志组下方创建多个日志流，方便对日志做进一步分类管理。创建日志流后，您可以通过将日志数据采集保存到日志流上，通过日志流对日志数据进行搜索分析、可视化展示、日志告警、日志转储、日志加工等。

为了让您更快了解并使用云日志服务LTS，建议您按照如下步骤进行操作：

1. 创建日志组，请参考**管理日志组**。
2. 创建日志流，请参考**管理日志流**。
3. 支持多账号日志汇聚多个账号的日志流、日志组集中管理。请参考**设置多账号日志汇聚**。
4. 在日志管理首页查看资源统计、我的收藏/我的收藏（本地缓存）、最近访问等信息。请参考**查看日志管理**。

3.2 管理日志组

日志组（LogGroup）是云日志服务进行日志管理的基本单位，用于对日志流进行分类，一个日志组下面可以创建多个日志流。日志组本身不存储任何日志数据，仅方便用户管理日志流，每个账号下可以创建100个日志组。

一个日志组通常对应公司内的某一个项目/业务，建议将某个项目/业务下的多个应用/服务的日志流归属到同一个日志组下。当公司有多个项目时，具体的项目人员只需要查看所属项目对应的日志组下的日志流即可，其它项目的日志流不会对其产生干扰。

LTS支持对日志组添加标签，按照业务需求对不同的日志组添加对应的标签，方便运维人员管理业务。

前提条件

如果您使用华为账号创建的IAM用户进行操作，IAM用户需要具备足够的权限才能使用云日志服务。具体操作请参见**授权IAM用户使用LTS**。

创建日志组

步骤1 登录**云日志服务控制台**。

步骤2 进入“日志管理”页面，单击“创建日志组”。

步骤3 在“创建日志组”页面中，参考**表3-1**填写日志组相关信息。

说明

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考**标签策略概述**，标签管理详细介绍请参考**标签管理**。

表 3-1 日志组参数说明

参数	说明
日志组名称	• 日志组名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。 • 日志采集后，将发送到对应的日志组中的日志流，如果日志较多，需要分门别类，建议您给日志组做好命名，方便后续快速查找日志。
企业项目	选择业务需要的企业项目，也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。
日志存储时间(天)	日志组的存储时间，即日志上报到LTS后日志存储的时间。 云日志服务LTS根据配置的日志存储时间定时清理日志内容，例如日志存储时间为30天，上报到LTS的日志只保存30天，30天后开始删除日志内容。 **说明** • 目前白名单用户的日志存储时间支持1095天，如有需要，请提工单申请。详细操作请参考**提交工单**。 • 创建日志组免费，使用阶段按照日志量收费，详细请参考**价格计算器**。
标签	按照业务需求对不同的日志组添加对应的标签。单击“添加标签”，分别填写标签键key和标签值value，开启应用到日志流。 **说明** • 如需添加多个标签可重复该步骤。 • 如需删除标签可单击标签操作列的 。 • 标签键长度不能超过128个字符；标签值长度不能超过255个字符。 • 标签键名称不可重复。 • 如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。
备注	自定义填写备注信息，字符长度0-1024个字符。

步骤4 单击“确定”，日志组创建成功，即可在日志组列表下方生成一条日志组信息。

- 在日志组列表中，可以查看日志组名称、标签、日志流数量等信息。
- 单击日志组名称，可跳转到日志流详情页面。
- 并发创建时，可能会偶现创建个数超过限制。

----结束

修改日志组

日志组创建完成后，如果您需要修改日志组的日志名称、日志存储时间或备注，参考如下操作：

步骤1 在日志组列表中，单击待修改日志组操作列下的“修改”。

步骤2 在弹出的修改日志组页面中，修改日志名称、日志存储时间。

步骤3 完成后，单击“确定”。

步骤4 修改成功后，鼠标悬浮在日志组名称上，显示修改后日志组名称和日志组原始名称。

----结束

删除日志组

如果日志组不再需要使用，可以删除日志组。日志组删除后，日志组中的日志流、日志数据将被同时删除。**日志组删除后无法恢复，请谨慎操作。**

说明

如果日志组绑定了日志转储任务，删除日志组之前，需要先删除该日志组关联的日志转储任务。

步骤1 在日志组列表中，单击待删除日志组操作列下的“删除”。

步骤2 在弹出框中输入“DELETE”后，单击“确定”，完成日志组删除。

图 3-1 删除日志组

----结束

搜索日志组/日志流

在日志组列表中，支持通过如下筛选条件进行搜索：

- 日志组/日志流
- 日志组原始名称/日志流原始名称
- 日志组名称/ID
- 日志流名称/ID
- 日志组标签
- 备注

其他操作

- 在日志组列表中，单击目标日志组操作列的“详情”，可查看该日志组名称、日志组ID、创建时间等详情。
- 需要先开启“ICAgent诊断开关”，请参考**设置ICAgent日志采集开关**。单击目标日志组操作列下的“更多>采集诊断”，可查看该日志组的ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。
- 单击搜索框旁边的 ，下载当前展示日志组的所有信息到本地查看。

3.3 管理日志流

云日志服务是以日志流作为日志管理维度，日志流（LogStream）是日志读写的基本单位，日志流是在日志组下方创建，将不同类型的日志分类存储在不同的日志流上，方便对日志进一步分类管理。例如，您可以将操作日志、访问日志等接入不同的日志流，查询日志时可以进入对应的日志流快速查看日志。支持按照业务需求对不同的日志流添加对应的标签，方便运维人员管理业务。

1个日志组中最多可以创建100个日志流，不支持扩大配额。如果您的配额已满，无法创建日志流，建议删除不再需要使用的日志流后重试，或者在新的日志组中创建日志流。

前提条件

已创建日志组。

创建日志流

步骤1 登录**云日志服务控制台**。

步骤2 单击日志组名称对应的 按钮。

步骤3 单击“创建日志流”，在创建日志流页面，参考**表3-2**填写日志流相关信息。

说明

若您的组织已经设定云日志服务的相关标签策略，则需按照标签策略规则为日志组、日志流、日志接入、主机组添加标签。标签如果不符合标签策略的规则，则可能会导致日志组、日志流、日志接入、主机组创建失败，请联系组织管理员了解标签策略详情。标签策略详细介绍请参考**标签策略概述**，标签管理详细介绍请参考**标签管理**。

表 3-2 日志流参数说明

参数	说明
日志组名称	默认显示目标日志组名称。
日志流名称	• 日志流名称只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。长度为1-64个字符。 • 日志采集后，以日志流为单位，将多条日志数据发往云日志服务。如果日志较多，需要分门别类，建议您创建多个日志流，并给日志流做好命名，方便后续快速查找日志。
企业项目	选择业务需要的企业项目，默认为default。也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。
日志存储	若关闭日志存储，则无法开启日志存储时间。 • 开启“日志存储”：日志将会被存入搜索引擎，能使用日志全量功能。 • 关闭“日志存储”：日志不会落盘存储，可节约索引流量和存储费用，只能使用日志生成指标、转储功能，不能使用日志搜索分析、告警、消费加工等其他功能。
日志存储时间(天)	日志流的存储时间，即日志上报到LTS后日志存储的时间。 • 打开日志流的“日志存储时间”：日志的存储时间使用日志流设置的日志存储时间。 • 关闭日志流的“日志存储时间”：日志的存储时间使用日志组设置的日志存储时间。
智能冷存储	开启日志存储时间后，根据业务需要设置**智能冷存储**功能。详细请参考**智能冷存储**。 **说明** 开启智能冷存储需要日志存储时间大于7天。
标准存储层数据保存时间	开启**智能冷存储**后，需要设置标准存储层数据的保存时间。 标准存储数据至少需保存7天才能转换为智能冷存储层数据。当日志保存时间大于**标准存储层数据保存时间**，且小于**日志存储时间**时，日志数据将自动转换为智能冷存储层数据存储。
标签	按照业务需求对不同的日志流添加对应的标签，单击添加标签，分别填写标签键key和标签值value。 **说明** • 如需添加多个标签可重复该步骤。 • 如需删除标签可单击标签操作列的  。 • 标签键长度不能超过128个字符；标签值长度不能超过255个字符。 • 标签键名称不可重复。 • 如果配置转储时使用了该标签，删除标签后，请同步修改转储配置信息。

参数	说明
匿名写入	匿名写入默认关闭，适用于安卓/IOS/小程序/浏览器端上报日志，打开匿名写入则表示该日志流打开匿名写入权限，不会经过有效鉴权，可能产生脏数据。关于SDK使用请参考**SDK接入**。
备注	自定义填写备注信息，字符长度0-1024个字符。

步骤4 单击“确定”，完成日志流的创建。在日志流列表中，可以查看日志流名称、操作等信息。

说明

- 支持查看日志流计费状态，日志计费请参考**价格计算器**。
- 按日志流维度上报话单功能目前在友好用户内测中，您可以**提交工单**申请开通。

----结束

智能冷存储

说明

目前此功能仅在华北-北京四、华南-广州、华东-上海一局点支持白名单用户**提交工单**申请使用，其他局点暂不支持该功能。

云日志服务LTS提供冷存储和标准存储功能。冷存储费用低，响应速度慢，对长期存储的数据进行冷存储。

- 在日志流创建或修改页面，开启智能冷存储功能，将标准存储层数据转为冷存储。
- 在开启智能冷存储功能后，如果标准存储时间增大，冷数据的存储时间仅对最新数据生效，冷存储数据不会重新转为标准存储。
- 在开启智能冷存储功能后，在可视化日志、仪表盘、定时SQL的查询时间范围查询日志时，不支持查询涉及冷存储的时间段，需将查询时间设置为标准存储的时间范围。

表 3-3 冷存储和标准存储区别

对比项	标准存储	冷存储
适用场景	数据高频查询场景	数据低频查询场景
查询与分析性能	高，秒级返回	较低，存在8S延迟
可视化、仪表盘等	高，秒级返回	不支持此功能
存储价格	0.000479元/GB/小时	0.000208元/GB/小时

修改日志流

日志流默认的存储时间和日志组保持一致。

步骤1 在日志流列表中，单击待修改日志流所在行的 按钮。

步骤2 在弹出框中，支持修改日志流名称、日志存储时间等信息。

说明

- 关闭开关：日志流会使用日志组配置的日志存储时间。
- 打开开关：使用日志流配置的日志存储时间。
- 超出存储时间的日志将会被自动删除，您可以按需将日志数据转储至OBS桶中进行长期存储。
- 目前白名单用户的日志存储时间支持1095天，如有需要，请**提交工单**申请。

步骤3 单击“确定”。

步骤4 修改完成后，鼠标悬浮在日志流名称上，显示修改后日志流名称和日志流原始名称。

----结束

删除日志流

如果日志流不再需要使用，可以删除日志流，日志流删除后，日志流中的日志数据将被同时删除。**日志流删除后无法恢复，请谨慎操作。**

说明

- 删除日志流前请确认该日志流下没有配置日志采集任务，否则删除后可能影响正常的日志上报。
- 如果日志流绑定了日志转储任务，删除日志流之前，需要先删除该日志流关联的日志转储任务。

步骤1 在日志流列表中，单击待删除日志流所在行的 。

步骤2 在弹出框中输入“DELETE”后，单击“确定”，完成日志流删除。

图 3-2 删除日志流

----结束

其他操作

- **收藏日志流**

  单击日志流中操作列下的 ，收藏日志流，即可在我的收藏/我的收藏（本地缓存）中展示已收藏的日志流。

- **详情**

  单击日志流中操作列下的 ，可查看日志流详情。包括日志流名称、日志流ID、创建时间等信息。

- 采集诊断：需要先开启“ICAgent诊断开关”，请参考**设置ICAgent日志采集开关**。单击目标日志流操作列下的 ，可查看该日志流的ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

3.4 查看日志管理

在日志管理首页提供资源统计、日志应用、我的收藏/我的收藏（本地缓存）、最近访问、告警统计、最新告警、功能上线公告和FAQ等信息的展示。

资源统计

在日志资源统计页面，支持查看资源统计、资源详情。日志资源统计是对日志进行分类统计及日志数据的可视化展示，统计日志资源的数据量仅供参考。

说明

标准存储量统计是指在选定时间范围内最新的存储总量。

步骤1 在云日志服务管理控制台，进入“日志管理”页面。

步骤2 在日志总览下方，单击资源统计旁边的相关明细，进入日志资源统计详情页面。

步骤3 资源统计主要展示日志资源数据，默认展示时间为1周（相对）的日志资源数据，您可以根据自己的实际需求选择时间范围。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

说明

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤4 在资源统计下方，查看资源统计数据：

- 读写流量：读写流量根据传输的流量计算，传输流量为压缩后的日志大小，日志一般有5倍压缩率。
- 索引流量-标准型日志流：原始日志数据默认都会建立全文索引，创建索引（对日志分词处理）后，才能搜索日志，在日志写入数据库时一次性收取流量费用。
- 标准存储量：日志标准存储量为压缩后的日志数据、索引数据、副本数据之和，这些空间约等于原始日志数据大小。

- 原始日志流量：原始日志数据的大小。

步骤5 在资源详情下方，查看Top100的日志组资源统计和Top100的日志流资源统计。根据选择的时间范围，支持表格或柱状图展示每日标准存储量（GB）、每日索引流量-标准型日志流（GB）和每日读写流量（GB）的数据。

- 新创建的日志组/日志流，需间隔至少1小时才能进行资源统计。
- 单击Top100中的日志组名称，可查询该日志组下的日志流资源统计。
- 单击  按钮，可下载日志组资源统计和日志流资源统计。

  说明

  下载的日志组资源统计和日志流资源统计文件为.CSV格式。

----结束

告警统计和最新告警

在日志总览下方，支持查看告警统计和最新告警。

- 告警统计展示云服务日志的告警总数及各个告警级别的数量。告警统计时间有：近30分钟、近1小时、近6小时、近1天和近1周；告警级别包括紧急、重要、次要和提示。
- 最新告警展示最新创建的告警规则，最多可显示近30分钟的3条告警规则。如需查看更多告警或添加告警，您可以单击 。

日志应用

在日志管理页面的日志总览下方，支持多种日志应用，即LTS提供开箱即用的日志仪表盘模板，用户接入即可进行日志的快速分析，主要有如下几个应用：

- ELB日志中心：支持ELB日志接入LTS和ELB仪表盘模板，详细请参考**ELB仪表盘模板**。
- APIG日志中心：支持APIG日志接入LTS和APIG仪表盘模板，详细请参考**APIG仪表盘模板**。
- VPC日志流中心：支持VPC日志接入LTS和VPC仪表盘模板，详细请参考**VPC仪表盘模板**。
- CFW日志中心：支持CFW日志接入LTS和CFW仪表盘模板，详细请参考**CFW仪表盘模板**。
- CTS日志中心：支持CTS日志接入LTS，暂不支持CTS仪表盘模板，详细请参考**CTS日志中心**。
- 多账号日志汇聚中心：支持您将多个账号下的日志流复制到指定的账号中，实现多账号日志的集中存储和分析。详细请参考**设置多账号日志汇聚**。

日志组列表

在日志组列表下方，展示日志组和日志流信息，更多信息请参考**管理日志组**、**管理日志流**。

购买资源包

云日志服务支持购买资源包进行计费，在费用结算时，优先从资源包中抵扣用量。详细请参考**购买LTS资源包**。

公告

公告是对新功能的介绍，展示云日志服务功能的最新动态。

如需查看更多功能介绍，您可以单击**更多**。

我的收藏/我的收藏（本地缓存）

我的收藏展示您收藏的日志流，有两种收藏方式：我的收藏和我的收藏（本地缓存）。

- **我的收藏**：将日志流保存至数据库中，默认为关闭状态。当您的账号开通写权限时，可显示该功能和我的收藏（本地缓存）。
- **我的收藏（本地缓存）**：将日志流保存至浏览器本地缓存，默认为关闭状态。所有账号均显示我的收藏（本地缓存）。

  说明

  当您的账号开通写权限时，**我的收藏/我的收藏（本地缓存）**至少有一个是开启状态，否则无法收藏日志流。

您可以通过云日志服务提供的收藏功能个性化定制属于自己的收藏日志流列表，方便您直接、快速的定位到常用的日志流。

以日志组lts-test为例，收藏日志组lts-test下某个日志流的操作步骤如下：

步骤1 在日志管理页面的日志组列表下方，单击日志组lts-test对应的 ，选择待收藏的日志流。

步骤2 单击日志流操作列的 图标，编辑收藏，选择收藏方式，单击“确定”，即可收藏日志流。

说明

编辑收藏取消已收藏的日志流，推荐如下两种方式：

- 在日志流列表中，单击待取消收藏的日志流对应的，即可取消收藏。
- 在我的收藏中，鼠标悬浮待取消收藏的日志流，单击，即可取消收藏。

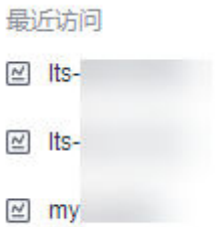
步骤3 收藏成功后，在右侧我的收藏/我的收藏（本地缓存）展示框中，即可展示您收藏的日志流信息。

----结束

最近访问

最近访问展示最近访问的日志流。最近访问最多可显示3条日志流访问记录。

图 3-3 最近访问



FAQ

FAQ（常见问题）展示经常被询问的问题。如需查看更多FAQ，您可以单击**更多**。

3.5 设置多账号日志汇聚

多账号日志汇聚中心支持您将多个账号下的日志流复制到指定的账号中，实现多账号日志的集中存储和分析，满足安全合规、集中分析等不同场景下的诉求。

说明

目前此功能支持华东-上海一、中国-香港、亚太-曼谷、亚太-新加坡、亚太-雅加达、华南-广州、拉美-圣保罗一、华北-北京四、西南-贵阳一、拉美-圣地亚哥，其他局点需要提交工单申请才能使用。详细操作请参考**提交工单**。

背景信息

- 集团公司经常采用多账号解决方案（LandingZone），不同的业务部门使用不同的账号，实现权限、资源等的隔离，提高账号的安全性。
- 集团公司的安全合规部门有统一收集日志的诉求，期望将不同账号下各个业务部门的关键日志集中存储和分析，汇聚到一个日志账号中，用于应对不同国家和地区的安全合规审计要求。
- 集团公司的运营部门也可能出于运营分析的诉求，期望将不同账号下各个业务部门的关键日志集中存储和分析，汇聚到一个日志账号中，方便后续进行统一的大数据处理和可视化展示。

方案介绍

- 组织（以下简称Organizations）云服务为企业用户提供多账号关系的管理能力。Organizations支持用户将多个华为云账号整合到创建的组织中，并可以集中管理组织下的所有账号。用户可以在组织中设置访问策略，帮助用户更好地满足业务的安全性和合规性需求。
- 云日志服务LTS联合Organizations推出多账号日志汇聚中心，您可以在Organizations使用管理账号指定某个成员账号成为LTS可信服务的委托管理员账号，然后可以在LTS将组织下某个成员账号的日志流复制到管理账号或者委托管理员账号中，实现多账号日志集中汇聚的目的。
- 某个成员账号的源日志组/日志流实际上是被复制一份到管理账号或者委托管理员账号对应的目标日志组/日志流中，因此两个账号间的日志组/日志流之间不会互相干扰，可以在各自账号下独立配置转储、消费、加工等任务。

前提条件

- 已创建组织。更多信息请参考**创建组织**。
- 已设置service.LTS为可信服务。更多信息请参考**启用、禁用可信服务**。
- 赋予管理账号或者委托管理员账号拥有Organizations服务只读权限。关于委托账号的操作请参考**添加、查看和取消委托管理员**。关于授权的操作请联系企业管理员参考**创建IAM用户并授权管理组织**。

设置多账号日志汇聚任务

步骤1 使用管理账号或者委托管理员账号登录云日志服务管理控制台，进入日志管理页面。

步骤2 在日志应用下方，单击多账号日志汇聚中心。

步骤3 在多账号日志汇聚配置页面，开启日志接收状态，左侧选择某个成员账号，右侧勾选对应的源日志组/日志流，支持自定义目标日志组/日志流的名称。

说明

- 若不需要使用日志汇聚配置功能时，可以关闭日志接收状态，则所有汇聚配置都会失效，源日志流停止汇聚到目标日志流上。
- 支持创建日志流的数量不超过总数量。更多信息请参考**基础资源**限制。

步骤4 设置完成后，单击“确定”，该账号汇聚配置创建中，预计5分钟左右创建完成，请稍后刷新查看配置。

说明

- 目标日志流初始化时默认采用源账号日志流的索引配置和结构化配置，配置成功后，若源账号日志流修改索引/结构化配置则不会同步到目标日志流。
- 删除源账号的日志组/日志流不会对目标账号的日志组/日志流造成影响。
- 取消勾选日志组/日志流并成功下发配置后，该日志组/日志流将不再继续汇聚。

----结束

4 日志接入

4.1 日志接入概述

云日志服务提供实时日志采集功能，支持ICAgent采集、云服务、API接入、SDK接入等日志采集方式，采集日志到LTS后，日志数据可以在云日志控制台以简单有序的方式展示、方便快捷的方式进行查询。

- **使用ICAgent插件采集日志**：云日志服务支持通过ICAgent采集方式进行日志上报。
- **使用云服务接入LTS**：云日志服务支持多个云服务日志接入，您可以选择不同云服务查看相应的日志接入方式。
- **使用API接入LTS**：云日志服务API提供了一系列上报日志方法，您可以查看使用API将日志接入云日志服务的方式。
- **使用SDK接入LTS**：云日志服务SDK提供了一系列上报日志方法，您可以查看使用SDK将日志接入云日志服务的方式。
- **跨IAM账号接入LTS**：通过创建委托，您可以将委托账号的日志流映射到被委托方账号，即就是将委托账号的日志流映射到当前云日志服务账号的日志流下。
- **使用KAFKA协议上报日志到LTS**：通过Kafka协议上报日志到日志服务，目前支持各类Kafka Producer SDK或采集工具，仅依赖于Kafka协议。

说明

日志接入前，需要确认开启ICAgent采集开关，请参考**设置LTS日志采集配额和使用量预警**。

- 采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。
- 日志采集关闭后，ICAgent会停止采集日志，且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。

4.2 使用 ICAgent 插件采集日志

4.2.1 安装 ICAgent（区域内主机）

ICAgent是云日志服务进行日志采集的工具，运行在需要采集日志的主机中。使用云日志服务在主机采集日志时，需要安装ICAgent。您可以通过以下操作指导在主机中安装ICAgent。

前提条件

安装ICAgent前，请确保本地浏览器的时间、时区与主机的时间、时区一致。如果不一致，可能会导致日志上报出错。

使用限制

- Linux环境：支持安装ICAgent的**Linux操作系统**。
- Windows环境：仅支持在如下64位系统的Windows环境中安装ICAgent。

```
Windows Server 2016 R2 Datacenter
Windows Server 2016 R2 Standard
Windows Server 2016 Datacenter English
Windows Server 2016 R2 Standard English

Windows Server 2012 R2 Datacenter
Windows Server 2012 R2 Standard
Windows Server 2012 Datacenter English
Windows Server 2012 R2 Standard English

Windows Server 2008 R2 Enterprise
Windows Server 2008 R2 Standard
Windows Server 2008 Enterprise English
Windows Server 2008 R2 Standard English
```

  Windows环境不支持在云日志服务主机管理界面对ICAgent进行升级和卸载操作，只支持日志采集功能。如果需要使用新版本，请先卸载旧版本ICAgent，再安装新版本ICAgent即可。

安装方式说明

ICAgent有两种安装方式，请按照您的场景进行选择。

表 4-1 安装方式

方式	适用场景
首次安装	该服务器上未安装过ICAgent。
继承安装（Linux环境支持）	您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用该安装方式。

首次安装（Linux 环境）

步骤1 在云日志服务管理控制台，单击“主机管理”。

步骤2 在主机管理页面，单击右上角“安装ICAgent”。

步骤3 “主机类型”选择“区域内主机”。

步骤4 “安装系统”选择“Linux”。

步骤5 选择“安装方式”。

- 获取AK/SK，方法请参考：**如何获取访问密钥AK/SK**

  请获取并使用公共用户账号的AK/SK，请勿使用个人账号的AK/SK。

  说明

  请确保公共用户账号及其创建的AK/SK不会被删除或禁用。AK/SK被删除，会导致安装的ICAgent无法正常上报数据到LTS。

- 当前支持通过**创建终端节点**和AOM/LTS服务打通网络，进行心跳和指标上报，以及日志上报。目前此功能仅支持西南-贵阳一局点，其他局点暂不支持。
- 创建IAM委托，方法请参考：**如何通过创建委托授权安装ICAgent**

步骤6 单击“复制命令”，复制ICAgent安装命令。

步骤7 使用PuTTY等远程登录工具，以**root**用户登录所在region待安装ICAgent的主机，执行ICAgent安装命令进行安装，当选择安装方式为“获取AK/SK”时需根据提示输入已获取到的AK/SK。

说明

- 当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在了/opt/oss/servicemgr/目录。安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器中ICAgent的状态。
- 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

----结束

首次安装（Windows 环境）

步骤1 在主机管理页面，单击右上角“安装ICAgent”。

步骤2 主机类型选择“区域内主机”。

步骤3 “安装系统”选择“Windows”。

步骤4 下载ICAgent安装包到Windows服务器。

您可以通过单击界面提供的ICAgent压缩包或者下载地址，下载ICAgent安装包。

步骤5 将ICAgent安装包存放到待安装ICAgent服务器的目录（如：C:\ICAgent）并解压。

步骤6 获取AK/SK方法请参考：**如何获取访问密钥AK/SK**

说明

如果AK/SK过期或者被删除，可能导致ICAgent状态显示异常。请创建新的AK/SK并生成的新的安装命令，登录到节点重新安装即可。

步骤7 手动替换AK/SK，单击“复制命令”，复制ICAgent安装命令。

步骤8 打开cmd窗口并进入ICAgent安装包的解压目录，执行ICAgent安装命令进行安装。

当显示“Service icagent installed successfully”时，表示安装成功。

说明

- 如果安装了第三方杀毒软件，需要添加ICAgent为信任程序，否则可能导致ICAgent安装失败。
- 如果需要卸载ICAgent，请在ICAgent安装包解压目录下，双击执行“*ICAgent安装包解压目录*\ICProbeAgent\bin\manual\win\uninstall.bat”脚本，当显示“icagent removed successfully”时，表示卸载成功。

  卸载ICAgent不会删除对应目录的文件，请您根据实际情况自行删除。
- 查询ICAgent的状态，**请在ICAgent安装包解压目录下**，打开cmd窗口，执行命令“sc query icagent”，状态为RUNNING，表示ICAgent正在运行中；提示“The specified service does not exist as an installed service”或者“指定的服务未安装”，表示ICAgent已卸载。
- **卸载后重新安装ICAgent，如果一直处于“pending”状态，可以在任务管理器中结束ICAgent相关进程，然后再次重新安装ICAgent。**

----结束

继承安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包**ICProbeAgent.tar.gz**，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承安装。

1. 在已安装ICAgent的服务器上执行如下命令，其中*x.x.x.x*表示待安装ICAgent服务器的IP地址。

   bash /opt/oss/servicemgr/ICAgent/bin/remoteInstall/remote_install.sh -ip x.x.x.x
2. 根据提示输入待安装ICAgent的服务器root用户密码。

   说明

   - 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成安装。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行安装。
   - 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待安装ICAgent的服务器进行远端通信。
   - 当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在了/opt/oss/servicemgr/目录。安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器ICAgent的状态。
   - 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

继承批量安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包**ICProbeAgent.tar.gz**，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承批量安装。

须知

- 批量安装的服务器需同属一个VPC下，并在同一个网段中。
- **批量安装功能依赖python3.*版本，如果安装时提示找不到python请安装python版本后重试。**

前提条件

已收集需要安装Agent的所有服务器的IP地址、密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的/opt/ICAgent/目录下。iplist.cfg格式示例如下所示，IP地址与密码之间用空格隔开：

192.168.0.109 密码（请根据实际填写）

192.168.0.39 密码（请根据实际填写）

说明

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，iplist.cfg中只需列出IP，无需填写密码，在执行时输入此密码即可；如果某个IP密码与其他不一致，则需在此IP后填写其密码。

操作步骤

1. 在已安装ICAgent的服务器上执行如下命令。

   bash /opt/oss/servicemgr/ICAgent/bin/remoteInstall/remote_install.sh -batchModeConfig /opt/ICAgent/iplist.cfg

   根据脚本提示输入待安装机器的root用户默认密码，如果所有IP的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。

   ```
   batch install begin
   Please input default passwd:
   send cmd to 192.168.0.109
   send cmd to 192.168.0.39
   2 tasks running, please wait...
   2 tasks running, please wait...
   2 tasks running, please wait...
   End of install agent: 192.168.0.39
   End of install agent: 192.168.0.109
   All hosts install icagent finish.
   ```

   请耐心等待，当提示All hosts install icagent finish.时，则表示配置文件中的所有主机安装操作已完成。

2. 安装完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看服务器的**查看ICAgent状态**。

4.2.2 安装 ICAgent（区域外主机）

ICAgent是云日志服务进行日志采集的工具，运行在需要采集日志的服务器中。使用云日志服务在区域外主机采集日志时，需要安装ICAgent。您可以通过以下操作指导在区域外主机中安装ICAgent。

前提条件

安装ICAgent前，请确保本地浏览器的时间、时区与主机的时间、时区一致。如果不一致，可能会导致日志上报出错。

使用限制

- Linux环境：支持安装ICAgent的**Linux操作系统**。
- Windows环境：仅支持在如下64位系统的Windows环境中安装ICAgent。

  ```
  Windows Server 2016 R2 Datacenter
  Windows Server 2016 R2 Standard
  Windows Server 2016 Datacenter English
  Windows Server 2016 R2 Standard English
  ```

```
Windows Server 2012 R2 Datacenter
Windows Server 2012 R2 Standard
Windows Server 2012 Datacenter English
Windows Server 2012 R2 Standard English

Windows Server 2008 R2 Enterprise
Windows Server 2008 R2 Standard
Windows Server 2008 Enterprise English
Windows Server 2008 R2 Standard English
```

Windows环境不支持在云日志服务主机管理界面对ICAgent进行升级和卸载操作，只支持日志采集功能。如果需要使用新版本，请先卸载旧版本ICAgent，再安装新版本ICAgent即可。

安装方式说明

ICAgent有两种安装方式，请按照您的场景进行选择。

表 4-2 安装方式

方式	适用场景
首次安装	该服务器上未安装过ICAgent。
继承安装（Linux环境支持）	您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，对于没有安装ICAgent的其他多个服务器，您可以采用该安装方式。

首次安装（Linux 环境）

非本区域的服务器安装ICAgent，请先在弹性云服务器控制台申请一台弹性云服务器作为跳板机，具体操作请参考**申请ELB和多主机作为跳板机**，然后执行如下操作：

说明

推荐CentOS 6.5 64bit及其以上版本的镜像， 最低规格为1vCPUs | 1GB，推荐规格为2vCPUs | 4GB。

步骤1 在当前region区申请一台ECS主机当跳板机。

步骤2 使用root用户登录跳板机，执行SSH Tunnel转发命令并修改跳板机ECS使用的安全组规则。

1. 在ECS详情页，单击安全组页签，进入安全组列表页。
2. 单击具体的安全组名，单击“更改安全组规则”，进入安全组详情页。
3. 在该安全组详情页，单击“入方向规则 > 添加规则 ”，按**表4-3**添加安全组规则。

表 4-3 安全组规则

方向	协议	端口	说明
入方向	TCP	8149、8102、8923、30200、30201、80	ICAgent发送数据到跳板机的端口列表。

说明

将安全组的入方向端口8149、8102、8923、30200、30201、80开启，保证非本区域的VM到跳板机ECS的数据连通性。

步骤3 在云日志服务管理控制台，单击“主机管理”。

步骤4 在主机管理页面，单击右上角“安装ICAgent”。

步骤5 “主机类型”选择“区域外主机”。

步骤6 “安装系统”选择“Linux”。

步骤7 设置“网络连通性方式”。云外K8S集群将日志上报到LTS的网络通路（公网或专线），建议您优先选择专线，更稳定可靠。

- 若选择公网，直接从**10**开始执行。
- 若选择专线，需要从步骤**8**开始执行。

步骤8 若网络连通性方式选择专线时，选择“连通LTS后端方式”为VPCEP时，填写VPCEP域名。

说明

- 专线连通场景下，默认其它IDC上的机器与LTS后端网络不通，需要配置网络打通方案，建议您选择VPCEP。
- 请您在华为云网络同事的协助下，在他云上配置VPCEP的DNS域名解析规则，将VPCEP的域名解析到指定IP；配置完成后，选择一台要采集日志的主机，按照 界面提示输入测试命令，如果能ping通，表示网络配置OK。

步骤9 若网络连通性方式选择专线时，选择“连通LTS后端方式”为跳板机时，需要设置该步骤。在跳板机上开通转发端口。

1. 输入跳板机私有IP，生成跳板机转发命令。

   说明

   跳板机私有IP是指VPC内网IP。

2. 单击“复制命令”，复制跳板机转发命令。
3. 以**root**用户登录跳板机，执行SSH Tunnel转发命令。

   ```
   ssh -f -N -L {跳板机ip}:8149:{elbip}:8149 -L {跳板机ip}:8102:{elbip}:8102 -L {跳板机ip}:8923:{elbip}:8923 -L {跳板机ip}:30200:{elbip}:30200 -L {跳板机ip}:30201:{elbip}:30201 -L {跳板机ip}:80:icagent-{region}.{obs_domain}:80 {跳板机ip}
   ```

   根据命令提示输入root用户密码即可。
4. 执行**netstat -lnp | grep ssh**命令查看对应端口是否被侦听，如果返回结果如**图4-1**所示，说明TCP端口已开通。

图 4-1 TCP 端口验证结果

说明

- 在浏览器地址栏里输入“http://跳板机ECS的IP地址”。如果访问成功，说明安全组规则已经生效。
- 如果跳板机ECS掉电重启，请重新执行如上命令。

5. **如何获取访问密钥AK/SK**，填写DC和跳板机连接IP。

说明

- DC：自定义节点所属数据中心名称，便于分类查看主机。
- 跳板机连接IP：使用EIP方式连接，为跳板机弹性公网IP，使用云专线VPC对等连接方式，为跳板机VPC内网IP。

步骤10 复制ICAgent安装命令。

步骤11 使用PuTTY等远程登录工具，以**root**用户登录所在region待安装ICAgent的服务器，执行ICAgent安装命令进行安装。

说明

- 当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在了/opt/oss/servicemgr/目录。安装成功后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器中ICAgent的状态。
- 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

----结束

首次安装（Windows 环境）

非本区域的服务器安装ICAgent，请先在弹性云服务器控制台申请一台Linux操作系统的弹性云服务器作为跳板机，具体操作请参考**申请ELB和多主机作为跳板机**，然后执行如下操作：

说明

推荐CentOS 6.5 64bit及其以上版本的镜像， 最低规格为1vCPUs | 1GB，推荐规格为2vCPUs | 4GB。

步骤1 在当前region区申请一台Linux操作系统的ECS主机当跳板机，修改跳板机ECS使用的安全组规则。

1. 在ECS详情页，单击安全组页签，进入安全组列表页。
2. 单击具体的安全组名，单击“更改安全组规则”，进入安全组详情页。
3. 在该安全组详情页，单击“入方向规则 > 添加规则 ”，按**表4-4**添加安全组规则。

表 4-4 安全组规则

方向	协议	端口	说明
入方向	TCP	8149、8102、8923、30200、30201、80	ICAgent发送数据到跳板机的端口列表。

说明

将安全组的入方向端口8149、8102、8923、30200、30201、80开启，保证非本区域的VM到跳板机ECS的数据连通性。

步骤2 在云日志服务管理控制台，单击“主机管理”。

步骤3 在主机管理页面，单击右上角“安装ICAgent”。

步骤4 主机类型选择“区域外主机”。

步骤5 “安装系统”选择“Windows”。

步骤6 设置“网络连通性方式”。云外Kubernetes集群将日志上报到LTS的网络通路（公网或专线），建议您优先选择专线，更稳定可靠。

- 若选择公网，直接从**9**开始执行。
- 若选择专线，需要从**7**开始执行。

步骤7 若网络连通性方式选择专线时，选择“连通LTS后端方式”为VPCEP时，填写VPCEP域名。

说明

- 专线连通场景下，默认其它IDC上的机器与LTS后端网络不通，需要配置网络打通方案，建议您选择VPCEP。
- 请您在华为云网络同事的协助下，在他云上配置VPCEP的DNS域名解析规则，将VPCEP的域名解析到指定IP；配置完成后，选择一台要采集日志的主机，按照 界面提示输入测试命令，如果能ping通，表示网络配置OK。

步骤8 若网络连通性方式选择专线时，选择“连通LTS后端方式”为跳板机时，需要设置该步骤。在跳板机上开通转发端口。

1. 输入跳板机私有IP，生成跳板机转发命令。

   说明

   跳板机私有IP是指VPC内网IP。

2. 单击“复制命令”，复制跳板机转发命令。
3. 以**root**用户登录跳板机，执行SSH Tunnel转发命令。

   ```
   ssh -f -N -L {跳板机ip}:8149:{elbip}:8149 -L {跳板机ip}:8102:{elbip}:8102 -L {跳板机ip}:8923:{elbip}:8923 -L {跳板机ip}:30200:{elbip}:30200 -L {跳板机ip}:30201:{elbip}:30201 -L {跳板机ip}:80:icagent-{region}.{obs_domain}:80 {跳板机ip}
   ```

   根据命令提示输入root用户密码即可。
4. 执行**netstat -lnp | grep ssh**命令查看对应端口是否被侦听，如果返回结果如图4-2所示，说明TCP端口已开通。

图 4-2 TCP 端口验证结果

说明

- 在浏览器地址栏里输入“http://跳板机ECS的IP地址”。如果访问成功，说明安全组规则已经生效。
- 如果跳板机ECS掉电重启，请重新执行如上命令。

步骤9 通过界面提示链接，下载ICAgent安装包。

步骤10 将ICAgent安装包存放到Windows主机目录（如：C:\ICAgent）并解压。

步骤11 获取AK/SK，请参考**如何获取访问密钥AK/SK**。

步骤12 若“连通LTS后端方式”为跳板机时，需要设置该步骤。生成安装命令，并复制该命令。

1. 在文本框输入跳板机连接IP，手动替换AK/SK，生成安装命令。

   说明

   跳板机连接IP：使用EIP方式连接，为跳板机弹性公网IP，使用云专线VPC对等连接方式，为跳板机VPC内网IP。

2. 单击“复制命令”，复制ICAgent安装命令。

步骤13 **打开cmd窗口并进入ICAgent安装包的解压目录**，执行ICAgent安装命令进行安装。

说明

- 当显示“Service icagent installed successfully”时，表示安装成功。安装成功后，在左侧导航栏中选择“主机管理 > 主机”，查看ICAgent状态。
- 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

----结束

申请 ELB 和多主机作为跳板机

说明

如果单跳板机有单点故障的风险，可以按照该方式提高接入的可靠性。

步骤1 创建跳板机弹性云服务器的Linux操作系统。

说明

根据业务规格配置CPU、内存，推荐规格为2vCPUs | 4GB以上。

步骤2 以root用户登录跳板机，使用该跳板机的内部IP，创建ssh通道。

1. 在弹性云服务器管理控制台找到已创建的机器，获取私有IP。

2. 在云日志管理控制台，依次单击“主机管理 > 安装ICAgent”，进入安装ICAgent页面。选择对应的操作系统，再选择主机类型为“区域外主机”，并填入私有IP，生成安装口令，登录跳板机，创建ssh通道。

步骤3 如果有多个跳板机，重复2，并且将多个跳板机放入同一个VPC中。在创建弹性云服务器，网络配置时，选择同一个虚拟私有云。

步骤4 创建ELB。具体请参见**创建弹性负载均衡**，在创建时需注意以下几点。

1. 创建ELB时选择与跳板机ECS相同的VPC。
2. 新建弹性公网IP，作为跳板机连接IP。
3. 带宽根据业务量申请，并进行适配。

步骤5 分别为TCP的端口30200、30201、8149、8923、8102添加监听器。具体请参见**添加TCP监听器**。

步骤6 为后端服务器组添加所有跳板机。具体请参见**添加后端云服务器**。

----结束

继承安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包**ICProbeAgent.tar.gz**，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承安装。

1. 在已安装ICAgent的服务器上执行如下命令，其中*x.x.x.x*表示待安装ICAgent服务器的IP地址。

   bash /opt/oss/servicemgr/ICAgent/bin/remoteInstall/remote_install.sh -ip x.x.x.x

2. 根据提示输入待安装ICAgent的服务器root用户密码。

   说明

   - 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成安装。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行安装。
   - 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待安装ICAgent的服务器进行远端通信。
   - 当显示“ICAgent install success”时，表示安装成功，ICAgent已安装在了/opt/oss/servicemgr/目录。安装成功后，在云日志服务左侧导航栏中选择主机管理 > 主机”，查看该服务器ICAgent的状态。
   - 如果安装失败，请卸载ICAgent后重新安装，如果还未安装成功，请联系技术支持。

继承批量安装（Linux 环境）

您有多个服务器需要安装ICAgent，其中一个服务器已经通过首次安装方式装好了ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent的安装包**ICProbeAgent.tar.gz**，对于没有安装ICAgent的服务器，可以通过该方式对服务器进行一键式继承批量安装。

须知

- 批量安装的服务器需同属一个VPC下，并在同一个网段中。
- **批量安装功能依赖python3.*版本，如果安装时提示找不到python请安装python版本后重试。**

前提条件

已收集需要安装Agent的所有服务器的IP地址、密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的/opt/ICAgent/目录下。iplist.cfg格式示例如下所示，IP地址与密码之间用空格隔开：

192.168.0.109 密码（请根据实际填写）

192.168.0.39 密码（请根据实际填写）

说明

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，iplist.cfg中只需列出IP，无需填写密码，在执行时输入此密码即可；如果某个IP密码与其他不一致，则需在此IP后填写其密码。

操作步骤

1. 在已安装ICAgent的服务器上执行如下命令。

   bash /opt/oss/servicemgr/ICAgent/bin/remoteInstall/remote_install.sh -batchModeConfig /opt/ICAgent/iplist.cfg

   根据脚本提示输入待安装机器的root用户默认密码，如果所有IP的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。

   ```
   batch install begin
   Please input default passwd:
   send cmd to 192.168.0.109
   send cmd to 192.168.0.39
   2 tasks running, please wait...
   2 tasks running, please wait...
   2 tasks running, please wait...
   End of install agent: 192.168.0.39
   End of install agent: 192.168.0.109
   All hosts install icagent finish.
   ```

   请耐心等待，当提示All hosts install icagent finish.时，则表示配置文件中的所有主机安装操作已完成。

2. 安装完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看服务器的**查看ICAgent状态**。

4.2.3 管理 ICAgent

ICAgent安装成功后，支持升级ICAgent、卸载ICAgent、查看ICAgent版本状态、查看ICAgent版本说明。

升级 ICAgent

为了更好的采集体验，LTS会不断更新ICAgent版本。当系统提示您有新的ICAgent版本时，您可以按照如下操作步骤进行升级。

说明

如果需要升级Windows环境中的ICAgent，请先卸载旧版本ICAgent，再安装新版本ICAgent即可。

步骤1 在云日志服务管理控制台，单击“主机管理”。

步骤2 在主机管理页面，选择“主机”页签。

步骤3 选择“区域内主机”，在主机列表中选中一个或多个待升级ICAgent前的复选框，单击“升级ICAgent”。

步骤4 选择“CCE集群”，在右侧下拉框中，选择待升级ICAgent的集群，单击“升级ICAgent”。

说明

- 未创建CCE集群时，采集容器标准输出到AOM的开关为置灰状态。
- 当ICAgent版本号为5.12.133及以上时，支持关闭采集容器标准输出到AOM的开关功能。
- 首次创建的CCE集群，默认集群下的主机已安装了ICAgent且上报日志到AOM，采集容器标准输出到AOM的开关处于开启状态；如需将日志上报至LTS则执行升级ICAgent操作时，关闭采集容器标准输出到AOM的开关。建议使用“接入日志 > 云服务接入 > 云容器引擎CCE”直接采集容器标准输出到LTS，不推荐采集到AOM。
- CCE集群ID（ClusterID）： 每个集群为固定的ID。
- 升级ICAgent时，LTS将为您的CCE集群创建对应的日志组和主机组。且该日志组和主机组的名称为k8s-log-{ClusterID}。您可以创建接入配置（云服务接入>云容器引擎CCE）将当前CCE集群的日志接入到该日志组。
- 当集群里的主机未安装ICAgent或ICAgent版本过低时，单击“升级ICAgent”操作，可对该集群里的所有主机安装ICAgent。

步骤5 在“升级ICAgent”对话框中单击“确定”。

ICAgent开始升级，升级ICAgent预计需要1分钟左右，请耐心等待。待ICAgent的状态由“升级中”变为“运行”时，表示升级成功。

说明

如果升级后，界面显示ICAgent状态异常或者其它升级失败场景，请直接登录节点使用安装命令重新安装ICAgent即可（覆盖式安装，无需卸载操作）。

----结束

卸载 ICAgent

服务器上的ICAgent被卸载后，会影响该服务器的日志采集能力，请谨慎操作！

说明

云日志服务主机管理界面，仅支持卸载安装在Linux环境中的ICAgent，如果需要卸载安装Windows环境中的ICAgent，请在ICAgent安装包解压目录下，双击执行“*ICAgent安装包解压目录*\ICProbeAgent\bin\manual\win\uninstall.bat”脚本，当显示“ICAgent uninstall success”时，表示卸载成功。

卸载ICAgent不会删除对应的安装文件，请您根据实际情况自行删除。

卸载方式，您可以按照需要进行选择：

- 通过界面卸载：此操作适用于正常安装ICAgent后需卸载的场景。

a. 在云日志服务管理控制台，单击“主机管理”，进入主机管理页面。
b. 单击“主机”切换至主机页签。
c. 勾选一个或多个待卸载ICAgent的服务器的复选框，单击“卸载ICAgent”。
d. 在“卸载ICAgent”对话框中单击“确定”。
ICAgent开始卸载，卸载ICAgent预计需要1分钟左右，请耐心等待。
卸载完成后主机列表中将不会显示该主机。

说明

通过界面卸载ICAgent后如果需要再次安装，请等待5分钟后执行安装操作，否则可能出现被再次自动卸载的情况。

- 登录服务器卸载：此操作适用于未成功安装ICAgent需卸载重装的场景。
  a. 以**root**用户登录需卸载ICAgent的服务器。
  b. 执行如下命令卸载ICAgent。
  bash /opt/oss/servicemgr/ICAgent/bin/manual/uninstall.sh;
  当显示“ICAgent uninstall success”时，表示卸载成功。
- 远程卸载：此操作适用于正常安装ICAgent后需远程卸载的场景。
  除了上述登录服务器上执行uninstall.sh命令卸载ICAgent的方式，还可以对服务器进行远程卸载。
  a. 在已安装ICAgent的服务器上执行如下命令，其中*x.x.x.x*表示待卸载ICAgent的服务器的IP地址。
  bash /opt/oss/servicemgr/ICAgent/bin/remoteUninstall/remote_uninstall.sh -ip x.x.x.x
  b. 根据提示输入待卸载ICAgent的服务器root用户密码。

  说明

  - 如果已安装ICAgent的服务器安装过expect工具，执行上述命令后，即可完成卸载。如果已安装ICAgent的服务器未安装expect工具，请根据提示输入密码，进行卸载。
  - 请确保已安装ICAgent的服务器可以使用root用户执行SSH、SCP命令，来与待卸载ICAgent的服务器进行远端通信。
  - 当显示“ICAgent uninstall success”时，表示卸载成功。
- 批量卸载：此操作适用于正常安装ICAgent后需批量卸载的场景。
  当您已有服务器安装过ICAgent，且该服务器“/opt/ICAgent/”路径下存在ICAgent安装包**ICProbeAgent.tar.gz**，通过该方式可对多个服务器进行一键式继承批量卸载。

须知

批量卸载的服务器需同属一个VPC下，并在同一个网段中。

前提条件

已收集需要卸载Agent的所有服务器的IP地址、密码，按照iplist.cfg格式整理好，并上传到已安装过ICAgent机器的/opt/ICAgent/目录下。iplist.cfg格式示例如下所示，IP地址与密码之间用空格隔开：

192.168.0.109 密码（请根据实际填写）

192.168.0.39 密码（请根据实际填写）

说明

- iplist.cfg中包含您的敏感信息，建议您使用完之后进行清理。
- 如果所有服务器的密码一致，iplist.cfg中只需列出IP地址，无需填写密码，在执行时输入此密码即可；如果某个IP密码与其他不一致，则需在此IP地址后填写其密码。

a. 在已安装ICAgent的服务器上执行如下命令。

bash /opt/oss/servicemgr/ICAgent/bin/remoteUninstall/remote_uninstall.sh -batchModeConfig /opt/ICAgent/iplist.cfg

根据脚本提示输入待卸载机器的root用户默认密码，如果所有IP地址的密码在iplist.cfg中已有配置，则直接输入回车键跳过即可，否则请输入默认密码。

```
batch uninstall begin
Please input default passwd:
send cmd to 192.168.0.109
send cmd to 192.168.0.39
2 tasks running, please wait...
End of uninstall agent: 192.168.0.109
End of uninstall agent: 192.168.0.39
All hosts uninstall icagent finish.
```

请耐心等待，当提示All hosts uninstall icagent finish.时，则表示配置文件中所有服务器的卸载操作已完成。

b. 卸载完成后，在云日志服务左侧导航栏中选择“主机管理 > 主机”，查看该服务器的ICAgent状态。

查看 ICAgent 状态

在主机管理页面的“主机”页签下方，查看目标主机的ICAgent状态。详细请参见表4-5。

表 4-5 ICAgent 状态

状态	说明
运行	该服务器的ICAgent运行正常。
未安装	该服务器未安装ICAgent。
安装中	正在为该主机安装ICAgent。安装ICAgent预计需要1分钟左右，请耐心等待。
安装失败	该主机的ICAgent安装失败。
升级中	正在升级该服务器的ICAgent。升级ICAgent预计需要1分钟左右，请耐心等待。
升级失败	该服务器的ICAgent升级失败。
离线	输入的AK/SK错误导致该主机的ICAgent功能异常。请获取正确的AK/SK后重新安装。
异常	该主机ICAgent功能异常，请联系技术支持。
卸载中	正在卸载该主机。卸载ICAgent预计需要1分钟左右，请耐心等待。
鉴权错误	安装该主机时配置的参数问题导致无法正常鉴权。

查看 ICAgent 版本说明

在主机管理页面的“主机”页签下方，查看目标主机的ICAgent版本。详细请参见表4-6。

表 4-6 ICAgent 版本说明

版本号	说明	发布时间
5.12.185	• 解决虚机日志配置中黑名单路径不生效问题。 • 优化containerd标准输出日志采集的问题。	2024-05-20
5.12.184	• 解决容器日志采集功能中无法排除绕接文件的问题。 • 节点日志采集功能并发采集协程调整为32个。	2024-05-16
5.12.183	优化containerd节点采集容器标准输出绕接文件的问题。	2024-05-11
5.12.182	解决syslog开关问题。	2024-04-28
5.12.181	• 解决自建k8s icagent认证失败问题。 • 解决日志截断问题。 • 解决日志速率很大的情况下，查找不到绕接文件导致文件漏采的问题。	2024-04-25
5.12.177	解决绕接死循环问题。	2024-03-28
5.12.176	• zip流式解析优化：检查转储文件是否结束。 • 限制podlb每个主机最大连接数。	2024-03-18
5.12.175	解决了结构化日志采集性能瓶颈问题。	2024-03-13
5.12.172	优化支持的绕接方式。	2024-02-28
5.12.171	解决Docker节点标准输出日志Json解析问题（没有去掉转义字符）。	2024-01-31
5.12.170	• 主机日志，容器日志，标准输出日志支持增量采集。 • 解决主机gpu指标挂断问题。	2024-01-29
5.12.166	• 解决标准输出日志采集插件占用内存高问题。 • 解决虚机日志采集插件重复采集绕接文件问题。 • 游标文件中添加日志组和日志流信息。	2023-12-27
5.12.165	从配置文件获取初始agentID，如果不符合校验要求则使用随机生成的uuid。	2023-12-21
5.12.163	支持UniAgent插件化安装ICAgent。	2023-12-13

版本号	说明	发布时间
5.12.159	● 解决标准输出日志采集协程泄露问题。 ● 解决标准输出日志采集到AOM后，不支持采集标准输出绕接日志的问题。	2023-11-27
5.12.158	解决关闭指标开关后容器指标内存泄露导致ICAgent重启的问题。	2023-11-08
5.12.157	● CCE接入LTS的容器日志采集：支持Docker驱动Devicemapper。 ● 解决虚机日志量大（转储快）ICAgent内存暴涨导致重启问题。	2023-11-06
5.12.156	解决从OBS拉取安装包问题，将http协议改成https。	2023-11-01
5.12.154	支持结构化功能。	2023-10-31
5.12.153	Release7版本。	2023-10-19
5.12.150	● 解决集群name和集群id not-set问题。 ● 支持CCE集群1.27版本。	2023-10-17
5.12.149	支持挂载绕接功能。	2023-10-12
5.12.148	修复gpu多卡场景，解决cpu高的问题。	2023-08-30
5.12.147	修复日志转储无法重启、主机gpu指标适配。	2023-08-17
5.12.142	支持CCE集群1.25及以上版本的容器gpu指标采集。	2023-06-13
5.12.139	解决上报LTS日志出现大量TIME_WAIT状态的TCP连接问题。	2023-04-25
5.12.135	● 解决CPU使用率为0的问题。 ● 解决CCE1.23版本集群containerd节点容器网络指标缺失问题。 ● 支持采集EulerOS 2.5系统的磁盘分区指标。	2023-02-08
5.12.133	容器的标准输出日志支持多行采集。	2022-12-17
5.12.130	支持将CCE日志直接接入LTS。	2022-11-04
5.12.120	● 增加进程的最大句柄数指标。 ● 支持LTS的podlb域名的切换能力。	2022-08-28
5.12.111	新增线程指标、修复“获取lvs磁盘分区指标失败”问题。	2022-06-09

版本号	说明	发布时间
5.12.100	• 上报内存指标增加内存workingset使用量、内存workingset使用率。 • 容器采集支持通过标签区分stderr.log和stdout.log。 • 容器上报增加Pod_ip的tag。 • **配置匹配当前目录文件。	2022-01-15
5.12.98	增加LTS日志黑名单功能，更改容器指标来源为working_set。	2021-09-29
5.12.96	新增云资源发现类型。	2021-09-22
5.12.90	更新gpu指标来源。	2021-07-15
5.12.87	新增磁盘支持类型。	2021-03-30
5.12.75	适配安全容器场景。	2021-03-09

4.2.4 管理 LTS 主机组

主机组是为了便于分类管理、提升配置多个主机日志采集的效率，对主机进行虚拟分组的单位。云日志服务支持通过一个接入配置来采集多台主机上的日志，您可以将这些主机加入到同一个主机组，并将该主机组关联至对应的接入配置中，方便您对多台主机日志进行采集。

- 当用户扩容主机时，只需在主机组中添加主机，该主机会自动继承关联的日志路径，无需为每台主机重复配置路径。
- 当用户修改多个主机采集路径时，只需修改对应的主机组关联的路径，无需为每台主机重复配置路径。

创建主机组（IP 地址）

1. 登录云日志服务管理控制台，单击“主机管理”，进入主机管理页面，单击右上角“新建主机组”。
2. 在弹出的新建主机组页面，输入“主机组名称”，主机组类型选择IP，主机类型选择“Linux主机”或“Windows主机”。

图 4-3 创建 IP 地址主机组

3. 在列表中选择需要加入该主机组的主机，单击“确定”，完成主机组的创建。
   - 可以通过主机名称或主机IP对列表进行过滤，也可以单击 批量搜索主机IP ，并在弹出的搜索框中输入多个主机IP，进行批量搜索。
   - 当列表中没有所需主机时，单击“安装ICAgent”，在弹出的页面安装指引完成主机安装，具体操作可参见**安装ICAgent（区域内主机）**或**安装ICAgent（区域外主机）**。

创建主机组（自定义标识）

1. 在主机管理页面，单击右上角“新建主机组”。
2. 在弹出的新建主机组页面，输入“主机组名称”，主机组类型选择“自定义标识”，主机类型选择“Linux主机”或“Windows主机”。

   说明

   单击“了解采集路径填写规则”，可查看详细的采集路径规则。

3. 单击 添加标识 ，添加自定义标识。

   说明

   最多可添加10个自定义标识。

4. 完成后，单击“确定”。
5. 执行以下操作创建custom_tag文件。
   a. 执行“cd /opt/cloud”命令，在cloud目录下，执行mkdir lts 创建lts目录。
   b. 继续执行“chmod 750 lts”，修改lts目录权限。
   c. 在lts目录下执行“touch custom_tag”，创建custom_tag文件 。
   d. 继续执行“chmod 640 custom_tag;vi custom_tag”命令，修改custom_tag权限并打开该文件。
   e. 按i进入insert模式，键入自定义标识后，按ESC键，“:wq!”保存退出即可。

说明

执行5之后，支持以下两种方式将主机加入到自定义标识主机组：

第一种（推荐使用）：

Linux主机

在主机里***/opt/cloud/lts***目录下的**custom_tag**文件中，查看该主机的标识，然后将该主机的标识，添加为主机组自定义标识，就可以将主机加入到该主机组下。例如：在主机里***/opt/cloud/lts***目录下的**custom_tag**文件中，查看该主机的标识为test1，创建主机组的自定义标识为test1，即将该主机加入到主机组下。

Windows主机

在主机里***C:|opt|cloud|lts***目录下的**custom_tag**文件中，查看该主机的标识，然后将该主机的标识，添加为主机组自定义标识，就可以将主机加入到该主机组下。例如：在主机里***C:|opt|cloud|lts***目录下的**custom_tag**文件中，查看该主机的标识为test1，创建主机组的自定义标识为test1，即将该主机加入到主机组下。

第二种：

Linux主机

- 在主机里***/opt/cloud/lts***目录下的**custom_tag**文件中，添加主机组自定义标识，可以将主机加入到该主机组下。例如：主机组的自定义标识为test，则在**custom_tag**文件中填写test，就可以将主机加入到该主机组下。
- 当添加了多个自定义标识时，在主机里***/opt/cloud/lts***目录下的**custom_tag**文件中，任意填写一个自定义标识，就可以将主机加入到该主机组下。

Windows主机

- 在主机里***C:|opt|cloud|lts***目录下的**custom_tag**文件中，添加主机组自定义标识，可以将主机加入到该主机组下。例如：主机组的自定义标识为test，则在**custom_tag**文件中填写test，就可以将将主机加入到该主机组下。
- 当添加了多个自定义标识时，在主机里***C:|opt|cloud|lts***目录下的**custom_tag**文件中，任意填写一个自定义标识，就可以将主机加入到该主机组下。

修改主机组

对于已创建的主机组可以对其名称进行修改，也可以对主机组进行添加主机、移除主机或者关联接入配置，具体操作如下：

表 4-7 操作列表

操作	具体步骤
修改主机组名称	1. 在主机管理页面，默认显示主机组页签。 2. 在主机组列表中，单击待修改的主机组所在行的操作列修改按钮。 3. 在弹出的修改主机组页面，修改主机组名称、自定义标识等信息。 4. 单击“确定”，完成主机名称修改。

操作	具体步骤
添加主机	**方式一：** 1. 在主机组列表，单击待修改的主机组类型为IP的主机组所在行前的 。 2. 在主机页签，单击“添加主机”。 3. 在弹出的添加主机页面，主机列表中显示该主机组所选主机类型下所有未选主机，选择需要加入该主机组的主机。 ● 可以通过主机名称或主机IP对列表进行过滤，也可以单击 批量搜索主机IP ，并在弹出的搜索框中输入多个主机IP，进行批量搜索。 ● 当列表中没有所需主机时，单击“安装ICAgent”，在弹出的页面安装指引完成主机安装，具体操作可参见**安装ICAgent**。 4. 单击“确定”。 **方式二：** 1. 在主机管理页面，单击“主机”，切换至主机页签。 2. 在主机列表中勾选需要添加的主机，单击“添加到主机组”。 3. 在弹出的添加到主机组页面，勾选目标主机组。 4. 单击“确定”，完成主机的添加。
移除主机	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 在主机页签，单击待移除主机所在行操作列的“移除”。 3. 在弹出的移除主机页面，单击“确定”，将该主机移除。 **说明** 自定义标识主机组下的主机不支持该操作。
取消部署	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 在主机页签，单击待移除主机所在行操作列的“取消部署”。 3. 在弹出的取消部署页面，单击“确定”，将该主机ICAgent卸载并移除。 **说明** ● 自定义标识主机组下的主机不支持该操作。 ● 主机取消部署后，其他主机组下的该主机也会被移除。
批量移除	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 在主机页签，勾选待移除的主机，单击“批量移除”。 3. 单击“确定”。

操作	具体步骤
新增关联配置	1. 在主机组列表，单击待修改的主机组所在行前的 。 2. 默认显示主机页签，单击“相关接入配置”，切换至相关接入配置页签。 3. 单击“新增关联配置”。 4. 在弹出的新增关联配置页面，勾选需要关联的接入配置。 5. 单击“确定”，配置完成后会将所选的接入配置显示在列表中。
解除关联	1. 在相关接入配置页签，单击待解除配置所在行操作列的“解除关联”。 2. 单击“确定”，解除该主机组与该接入配置的关联。
批量解除关联	1. 在相关接入配置页签，勾选待解除的配置，单击“批量解除关联”。 2. 单击“确定”，解除该主机组与所勾选的接入配置的关联。
复制主机组ID	鼠标悬浮在主机组名称上，支持复制主机组ID。
导出主机信息	1. 在主机页签的区域内主机、CCE集群或区域外主机下方，勾选需要导出的主机。 2. 单击“导出”，即可将主机信息导出到本地进行查看。

删除主机组

删除主机组

1. 在主机管理页面，默认显示主机组页签。
2. 在主机组列表中，单击待删除的主机组所在行的操作列删除图标。
3. 在弹出的删除主机组页面，单击“确定”，删除该主机组。

批量删除主机组

1. 在主机组列表，勾选待删除的主机组，单击列表左上方“批量删除”。
2. 在弹出的删除主机组页面，单击“确定”，删除所勾选的主机组。

4.2.5 裸金属服务 BMS 文本日志接入 LTS

裸金属服务器（Bare Metal Server）是一款兼具虚拟机弹性和物理机性能的计算类服务，为您和您的企业提供专属的云上物理服务器，为核心数据库、关键应用系统、高性能计算、大数据等业务提供卓越的计算性能以及数据安全。

- **创建单个接入配置**：创建裸金属服务器（BMS）文本日志接入LTS的单个接入配置任务。
- **创建多个接入配置**：同时创建裸金属服务器（BMS）文本日志接入LTS的多个接入配置任务。

前提条件

已**安装ICAgent**并**添加至主机组**。开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

创建单个接入配置的操作步骤

云日志服务接入方式选择裸金属服务 BMS - 文本日志时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”。在接入向导页签或者在接入规则页签单击“接入日志”，单击“裸金属服务 BMS - 文本日志”进行主机接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“裸金属服务 BMS - 文本日志”进行主机接入配置。

步骤4 选择日志组。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步：选择主机组（可选）”。

步骤5 选择主机组。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组，具体可参考**创建主机组（IP地址）**。

   说明

   主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。

   - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
   - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。
2. 单击“下一步：采集配置”。

步骤6 采集配置。

对主机日志采集设置具体的采集规则，具体请参考**采集配置**。

步骤7 结构化配置（可选项）。

结构化配置，具体请参考**设置云端结构化解析日志**。

说明

当所选日志流已配置结构化时，请谨慎执行删除操作。

若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

步骤8 索引配置（可选项）。

索引配置，具体请参考**索引配置**。

步骤9 单击“提交”，接入成功后，在接入规则页签，则会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。
- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用主机接入完成日志接入时，采集配置的具体配置如下：

1. 采集配置名称：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。

   说明

   导入旧版配置：将旧版主机接入配置导入到新版日志接入中。

   - 若是新安装云日志服务的场景，页面没有显示“导入旧版配置”，则表示不需要导入旧版配置，直接新建配置即可。
   - 若是升级云日志服务的场景，页面显示“导入旧版配置”，若需要旧版配置里的主机日志路径，可以选择导入旧版配置，或者直接新建配置。
   - 页面显示“导入其他配置”，则表示支持导入其他配置，可以选择已配置好的直接导入，不用重新配置。

2. 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。
   - **采集路径支持递归路径，**表示递归5层目录。**

     示例：采集路径配置为 /var/logs/**/a.log，日志匹配如下：

     ```
     /var/logs/1/a.log
     /var/logs/1/2/a.log
     /var/logs/1/2/3/a.log
     /var/logs/1/2/3/4/a.log
     /var/logs/1/2/3/4/5/a.log
     ```

     说明

     - 以上示例中的/1/2/3/4/5/，表示/var/logs目录中，往里递归的5个目录层级，在这5个目录层级中只要存在a.log，都能进行日志匹配。
     - 采集路径中只能出现一次**，不能出现两个及以上。正确示例：/var/logs/**/a.log；错误示例：/opt/test/**/log/**。
     - 采集路径中第一个层级不允许为**（避免误采集系统文件），错误示例：/**/test。
   - **采集路径支持模糊匹配，匹配目录或文件名中的任何字符**。

说明

如果配置了C:\windows\system32类似的日志采集路径，但无法采集日志，请尝试打开WAF物理防火墙后重新配置。

- 示例1：采集路径配置为 /var/logs/*/a.log，表示/var/logs/目录下，任何一个目录中存在a.log，都能进行日志匹配，例如：

  /var/logs/1/a.log

  /var/logs/2/a.log

- 示例2：采集路径配置为 /var/logs/service-*/a.log，日志匹配示例：

  /var/logs/service-1/a.log

  /var/logs/service-2/a.log

- 示例3：采集路径配置为 /var/logs/service/a*.log，日志匹配示例：

  /var/logs/service/a1.log

  /var/logs/service/a2.log

– **采集路径如果配置的是目录，示例：/var/logs/，则只采集目录下后缀为“.log”、“.trace”和“.out”的文件**。

  如果配置的是文件名，则直接采集对应文件，只支持内容是文本格式的文件。可以通过*file -i 文件名*命令，查询文件格式。

– 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。

  内置类型为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。

  自定义绕接规则为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log

说明

- 请注意您的敏感信息是否在收集范围内。
- 当主机选择“Windows主机”时，如需采集系统日志，需要在“采集配置”环节，开启“采集Windows事件日志”。
- windows事件日志采集不能重复配置，即相同主机下，即使跨日志组和日志流，也只能配置一次。
- LTS暂不支持采集PostgreSQL（数据库）实例的日志，目前只支持采集安装在ECS（主机）实例的日志。
- **日志采集路径不能重复配置**，即相同主机的同一个日志采集路径不能重复配置，否则可能会导致日志采集异常。
- 相同主机的同一个日志采集路径，如果在AOM进行了配置，则不能在LTS重复配置。
- 配置采集的文件最后修改时间和当前时间差如果已超过12小时，则不会采集。

3. 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件，但是不能过滤该目录下文件夹里的日志文件。

   目录和文件名支持完全匹配，也支持模糊匹配，具体可参考**路径配置内容**进行设置。

说明

- 当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。
- 已经加了黑名单的日志，新建日志接入也无法采集黑名单里的日志，除非在设置采集黑名单下方删除采集路径，才能重新采集。

4. 采集Windows事件日志：当选择Windows主机采集日志时，需要开启“采集Windows事件日志”，配置如下参数：

表 4-8 采集 Windows 事件日志参数

名称	说明
日志类型	日志类型有系统、应用程序、安全和启动。
首次采集时间偏移量	如设置为7天，表示从采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。
事件等级	事件等级有information、warning、error、critical和verbose。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。

5. 其他配置。

表 4-9 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启“日志拆分”，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启“日志拆分”，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。

名称	说明
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 ● 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 ● 全量采集：ICAgent采集新文件时，从文件的开头开始读。
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

6. 日志格式、日志时间具体说明如下：

表 4-10 日志采集信息

名称	说明
日志格式	● 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 ● 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** ● 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 ● 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 ● 采集日志时间限制：系统时间的前后24小时内。

名称	说明
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 ● 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 ● 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond (999) hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

说明

时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。**如果没有特殊需求，建议使用单行日志-系统时间模式即可**。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考表4-11。

表 4-11 批量接入设置

类型	操作	说明
基本配置	接入类型	选择裸金属服务BMS-文本日志。

类型	操作	说明
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考**创建单个接入配置的操作步骤**进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.6 云容器引擎 CCE 应用日志接入 LTS

支持云容器引擎（Cloud Container Engine）日志接入LTS。

- **创建单个接入配置**：创建云容器引擎 CCE-应用日志接入LTS的单个接入配置任务。
- **创建多个接入配置**：同时创建云容器引擎 CCE-应用日志接入LTS的多个接入配置任务。

前提条件

- CCE集群已安装ICAgent并且已创建相关节点自定义标识的主机组（如果不满足，配置CCE接入LTS时会自动检查修复）。
- 已**关闭采集容器标准输出到AOM的开关**。
- 开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

使用限制

- 支持容器引擎为Docker的CCE集群节点。详情请查看**云容器引擎（CCE）**。
- 支持使用Containerd作为容器引擎的CCE集群节点（ICAgent 5.12.130及以上版本）。
- 支持CCE Turbo集群（ICAgent 5.12.130及以上版本）。

- 容器内的日志目录如果已挂载到主机目录上，将无法通过配置容器文件路径方式采集到LTS，只能通过配置节点文件路径方式采集到LTS。
- Docker存储驱动限制：容器文件日志采集目前仅支持overlay2存储驱动，不支持devicemapper作为存储驱动的节点。查看存储驱动类型，请使用如下命令：
  ```
  docker info | grep "Storage Driver"
  ```
- 如果选择日志流时，采集方式为采集到集中日志流时，则必须已创建CCE集群。

创建单个接入配置的操作步骤

云日志服务接入方式选择CCE接入时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，在接入向导页签或者在接入规则页签单击“接入日志”，单击“云容器引擎 CCE-应用日志”进行CCE接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“云容器引擎 CCE-应用日志”进行CCE接入配置。

步骤4 选择日志流。

有两种采集方式：采集到自定义日志流和采集到集中日志流，您可以根据实际情况选择采集方式，推荐您使用采集到集中日志流。

采集到自定义日志流

1. 单击“CCE集群”后的目标框，在下拉列表中选择具体的集群。
2. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
3. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
4. 单击“下一步：检查依赖项”。

采集到集中日志流

集中采集日志到一个固定的日志流。CCE集群默认的采集日志流分别为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、K8S事件: event-{ClusterID}和容器文件containerfile-{ClusterID}。日志流名称会根据ClusterID自动命名，例如：集群ID为Cluster01，则标准输出/错误日志流为stdout-Cluster01。

在一个CCE集群下可以创建的采集日志流为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、容器文件containerfile-{ClusterID}）和K8s事件event-{ClusterID}，如果某个日志组下，已创建某种采集日志流，则不会在其他日志组或当前日志组下再创建该日志流。

1. 单击“CCE集群”后的目标框，在下拉列表中选择具体的集群。
2. 默认所属日志组为k8s-log-集群ID，例如集群ID为c7f3f4a5-bcb8-11ed-a4ec-0255ac100b07，默认所属日志组为k8s-log-c7f3f4a5-bcb8-11ed-a4ec-0255ac100b07。

说明

当无该日志组时，系统会提示：暂无该日志组，后续操作中，系统将会为您自动创建，创建完成后日志会集中采集到该日志组中。

3. 单击“下一步：检查依赖项”。

步骤5 检查依赖项。

系统自动检查以下内容检查项是否符合要求：

1. 已安装ICAgent，且版本 >= 5.12.130。
2. 存在名称和自定义标识都是**k8s-log-集群ID**的主机组。
3. 存在名为**k8s-log-集群ID**的日志组。
4. 存在系统推荐的集中采集的日志流。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。

如果以上内容检查项中，有任意一项不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。

说明

- **自动修复**：一键帮您完成以上内容检查项配置。
- **重新检查**：重新检查依赖项。
- 当选择日志流为**采集到自定义日志流**时，“存在名为**k8s-log-集群ID**的日志组”的检查项为可选项。您可以通过开启或关闭开关进行控制，确定是否进行该项检查。

步骤6 选择主机组（可选）。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组，具体可参考**创建主机组（自定义标识）**。

   说明

   - 默认选择集群所在的主机组，您可以根据需要选择其他已创建的主机组。
   - 主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。
     - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
     - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。
2. 单击“下一步：采集配置”。

步骤7 采集配置。

设置具体的采集规则，具体可参考**采集配置**。

步骤8 结构化配置（可选项）。

结构化配置，具体请参考**设置云端结构化解析日志**。

说明

当所选日志流已配置结构化时，请谨慎执行删除操作。

若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

步骤9 索引配置（可选项）。

索引配置，具体请参考**索引配置**。

步骤10 单击“提交”，完成CCE接入，在接入规则页签，则会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。
- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用CCE接入完成日志接入时，在采集配置页面的具体配置如下：

1. **基本配置**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置。
   - 容器标准输出：采集集群内指定容器日志，仅支持Stderr和Stdout的日志。

     说明

     - 被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。
     - 容器标准输出不能重复配置，即使跨日志组和日志流，也只能配置一次。
   - 容器文件路径：采集集群内指定容器内的文件日志。
   - 节点文件路径：采集集群内指定节点的文件。

     说明

     采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。
   - K8S事件：采集K8S集群内的事件日志。

     说明

     K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。

表 4-12 采集配置参数表

类型	参数配置
容器标准输出	• 采集容器标准输出到AOM：默认集群下的主机已安装了ICAgent且采集日志到AOM，采集容器标准输出到AOM的开关处于开启状态。开启后标准输出只会采集到AOM，不会采集到LTS，建议您手动关闭该开关。 • 采集容器标准输出（stdout）和采集容器标准错误（stderr）两者必须得有一个是开启状态。 • 开启采集容器标准错误（stderr）后，选择采集目前路径：将标准输出和标准错误采集到不同的文件（stdout.log和stderr.log）、将标准输出和标准错误采集到同一个文件（stdout.log）。
容器文件路径	• **路径配置**：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** • 当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。 • 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。 **内置类型**为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为**-._**绕接标识为非字母符号，后缀为字母。 **自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。

类型	参数配置
节点文件路径	• **路径配置**：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。 **内置类型**为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。 **自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。
K8S事件	无需设置参数。仅支持icagent 5.12.150及以上版本。

3. K8s匹配规则：当数据源类型选择容器标准输出和容器文件路径时，设置K8s匹配规则，非必选项。

   说明

   填写正则匹配规则后，单击校验按钮，支持校验确保正则表达式的正确性。

表 4-13 K8s 匹配规则

参数名称	参数说明
K8s Namespace正则匹配	通过Namespace名称指定采集的容器，支持正则匹配。 **说明** 采集名称符合正则规则的Namespace的日志，为空时采集所有Namespace的日志。
K8s Pod正则匹配	通过Pod名称指定待采集的容器，支持正则匹配。 **说明** 采集名称符合正则规则的Pod的日志，为空时采集所有Pod的日志。
K8s容器名称正则匹配	通过容器名称指定待采集的容器（Kubernetes容器名称是定义在spec.containers中），支持正则匹配。 **说明** 采集名称符合正则规则的容器的日志，为空时采集所有容器的日志。

参数名称	参数说明
K8s Label白名单	通过K8s Label白名单指定待采集的容器。如果您要设置K8s Label白名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则K8S Label中包含LabelKey的容器都匹配；若LabelValue不为空，则K8S Label中包含LabelKey=LabelValue的容器才匹配；LabelKey需要全匹配，LabelValue支持正则匹配；多个白名单之间为或关系，即只要K8S Label满足任一白名单即可被匹配。
K8s Label黑名单	通过K8s Label黑名单排除不采集的容器。如果您要设置K8s Label黑名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则K8S Label中包含LabelKey的容器都被排除；若LabelValue不为空，则K8S Label中包含LabelKey=LabelValue的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配；多个黑名单之间为或关系，即只要K8S Label满足任一黑名单即可被排除。
K8s Label日志标签	设置K8s Label日志标签后，日志服务将在日志中新增K8s Label相关字段。 **说明** 设置K8s Label日志标签后，lts将在日志中新增相关字段。例如设置LabelKey为app，设置LabelValue为app_alias，当容器中包含app=lts时，将在日志中添加内容{app_alias：lts}。
容器Label白名单	通过容器Label白名单指定待采集的容器。如果您要设置容器Label白名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则容器 Label中包含LabelKey的容器都匹配；若LabelValue不为空，则容器 Label中包含LabelKey=LabelValue的容器才匹配；LabelKey需要全匹配，LabelValue支持正则匹配；多个白名单之间为或关系，即只要容器 Label满足任一白名单即可被匹配。
容器Label黑名单	通过容器Label黑名单排除不采集的容器。如果您要设置容器Label黑名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则容器 Label中包含LabelKey的容器都被排除；若LabelValue不为空，则容器 Label中包含LabelKey=LabelValue的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配；多个黑名单之间为或关系，即只要容器 Label满足任一黑名单即可被排除。
容器Label日志标签	设置容器Label日志标签后，日志服务将在日志中新增容器Label相关字段。 **说明** 设置容器 Label日志标签后，lts将在日志中新增相关字段。例如设置LabelKey为app，设置LabelValue为app_alias，当容器中包含app=lts时，将在日志中添加的内容{app_alias：lts}。

参数名称	参数说明
环境变量白名单	用于指定待采集的容器。如果您要设置环境变量白名单，那么Label Key必填，Label Value可选填。 **说明** 如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都匹配；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才被匹配；LabelKey需要全匹配，LabelValue支持正则匹配；多个白名单之间为或关系，即只要容器的环境变量满足任一键值对即可被匹配。
环境变量黑名单	用于排除不采集的容器。如果您要设置环境变量黑名单，那么Label Key必填，Label Value可选填。 **说明** 如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都将被排除；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配；多个黑名单之间为或关系，即只要容器的环境变量满足任一键值对即可被排除。
环境变量日志标签	设置环境变量日志标签后，日志服务将在日志中新增环境变量相关字段。 **说明** 设置环境变量日志标签后，lts将在日志中新增相关字段，例如设置环境变量Key为app，设置环境变量Value为app_alias，当容器中包含环境变量app=lts时，将在日志中添加的内容为{app_alias：lts}。

4. 开启结构化解析配置，详细操作请参考**ICAgent结构化解析规则说明**。

   需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

   说明

   若已经配置了云端结构化解析，请先删除云端结构化解析后再配置ICAgent结构化解析。

   图 4-4 ICAgent 结构化解析配置

5. 其他配置。

表 4-14 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启日志拆分按钮，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启日志拆分按钮，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 ● 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 ● 全量采集：ICAgent采集新文件时，从文件的开头开始读。
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

6. 日志格式、日志时间具体说明如下：

说明

不再推荐使用以下功能，建议使用**结构化解析配置**。
如果您配置了ICAgent的多行全文或者多行完全正则，此处的多行日志配置会失效。

表 4-15 日志采集信息

名称	说明
日志格式	• 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 • 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** • 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 • 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 • 采集日志时间限制：系统时间的前后24小时内。
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 • 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 • 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond（999） hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

说明

时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。**如果没有特殊需求，建议使用单行日志-系统时间模式即可**。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考**表4-16**。

表 4-16 批量接入设置

类型	操作	说明
基本配置	接入类型	选择云容器引擎 CCE-应用日志。
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考**创建单个接入配置的操作步骤**进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.7 云主机 ECS 文本日志接入 LTS

当您选择了ECS接入方式时，云日志服务可以将ECS待采集日志的路径配置到日志流中，ICAgent将按照日志采集规则采集日志，并将多条日志进行打包，以日志流为单位发往云日志服务，您可以在云日志服务控制台实时查看日志。

- **创建单个接入配置**：创建云主机 ECS-文本日志接入LTS的单个接入配置任务。
- **创建多个接入配置**：同时创建云主机 ECS-文本日志接入LTS的多个接入配置任务。

前提条件

已**安装ICAgent**并**添加至主机组**。开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

创建单个接入配置的操作步骤

云日志服务接入方式选择**云主机 ECS-文本日志**时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，在接入向导页签或者在接入规则页签单击“接入日志”，单击“云主机 ECS-文本日志”进行主机接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“云主机 ECS-文本日志”进行主机接入配置。

步骤4 选择日志组。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步：选择主机组（可选）”。

步骤5 选择主机组。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组，具体可参考**创建主机组（IP地址）**。

   说明

   主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。

   - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
   - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。

2. 单击“下一步：采集配置”。

步骤6 采集配置。

对主机日志采集设置具体的采集规则，具体请参考**采集配置**。

步骤7 结构化配置（可选项）。

结构化配置，具体请参考**设置云端结构化解析日志**。

说明

当所选日志流已配置结构化时，请谨慎执行删除操作。

若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

步骤8 索引配置（可选项）。

索引配置，具体请参考**索引配置**。

步骤9 单击“提交”，接入成功后，在接入规则页签，则会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。
- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用主机接入完成日志接入时，采集配置的具体配置如下：

1. 采集配置名称：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。

   说明

   导入旧版配置：将旧版主机接入配置导入到新版日志接入中。

   - 若是新安装云日志服务的场景，页面没有显示“导入旧版配置”，则表示不需要导入旧版配置，直接新建配置即可。
   - 若是升级云日志服务的场景，页面显示“导入旧版配置”，若需要旧版配置里的主机日志路径，可以选择导入旧版配置，或者直接新建配置。
   - 页面显示“导入其他配置”，则表示支持导入其他配置，可以选择已配置好的直接导入，不用重新配置。

2. 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。
   - **采集路径支持递归路径，**表示递归5层目录。**

     示例：采集路径配置为 /var/logs/**/a.log，日志匹配如下：

     ```
     /var/logs/1/a.log
     /var/logs/1/2/a.log
     /var/logs/1/2/3/a.log
     /var/logs/1/2/3/4/a.log
     /var/logs/1/2/3/4/5/a.log
     ```

     说明

     - 以上示例中的/1/2/3/4/5/，表示/var/logs目录中，往里递归的5个目录层级，在这5个目录层级中只要存在a.log，都能进行日志匹配。
     - 采集路径中只能出现一次**，不能出现两个及以上。正确示例：/var/logs/**/a.log；错误示例：/opt/test/**/log/**。
     - 采集路径中第一个层级不允许为**（避免误采集系统文件），错误示例：/**/test。
   - **采集路径支持模糊匹配，匹配目录或文件名中的任何字符**。

说明

如果配置了C:\windows\system32类似的日志采集路径，但无法采集日志，请尝试打开WAF物理防火墙后重新配置。

- 示例1：采集路径配置为 /var/logs/*/a.log，表示/var/logs/目录下，任何一个目录中存在a.log，都能进行日志匹配，例如：

  /var/logs/1/a.log

  /var/logs/2/a.log

- 示例2：采集路径配置为 /var/logs/service-*/a.log，日志匹配示例：

  /var/logs/service-1/a.log

  /var/logs/service-2/a.log

- 示例3：采集路径配置为 /var/logs/service/a*.log，日志匹配示例：

  /var/logs/service/a1.log

  /var/logs/service/a2.log

– **采集路径如果配置的是目录，示例：/var/logs/，则只采集目录下后缀为“.log”、“.trace”和“.out”的文件**。

  如果配置的是文件名，则直接采集对应文件，只支持内容是文本格式的文件。可以通过*file -i 文件名*命令，查询文件格式。

– 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。

  内置类型为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。

  自定义绕接规则为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log

说明

- 请注意您的敏感信息是否在收集范围内。
- 当主机选择“Windows主机”时，如需采集系统日志，需要在“采集配置”环节，开启“采集Windows事件日志”。
- windows事件日志采集不能重复配置，即相同主机下，即使跨日志组和日志流，也只能配置一次。
- LTS暂不支持采集PostgreSQL（数据库）实例的日志，目前只支持采集安装在ECS（主机）实例的日志。
- **日志采集路径不能重复配置**，即相同主机的同一个日志采集路径不能重复配置，否则可能会导致日志采集异常。
- 相同主机的同一个日志采集路径，如果在AOM进行了配置，则不能在LTS重复配置。
- 配置采集的文件最后修改时间和当前时间差如果已超过12小时，则不会采集。

3. 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件，但是不能过滤该目录下文件夹里的日志文件。

   目录和文件名支持完全匹配，也支持模糊匹配，具体可参考**路径配置内容**进行设置。

说明

- 当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。
- 已经加了黑名单的日志，新建日志接入也无法采集黑名单里的日志，除非在设置采集黑名单下方删除采集路径，才能重新采集。

4. 采集Windows事件日志：当选择Windows主机采集日志时，需要开启“采集Windows事件日志”，配置如下参数：

表 4-17 采集 Windows 事件日志参数

名称	说明
日志类型	日志类型有系统、应用程序、安全和启动。
首次采集时间偏移量	如设置为7天，表示从采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。
事件等级	事件等级有information、warning、error、critical和verbose。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。

5. 开启结构化解析配置，详细操作请参考**ICAgent结构化解析规则说明**。

需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

说明

若已经配置了云端结构化解析，请先删除云端结构化解析后再配置ICAgent结构化解析。

图 4-5 ICAgent 结构化解析配置

6. 其他配置。

表 4-18 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启日志拆分按钮，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启日志拆分按钮，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 ● 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 ● 全量采集：ICAgent采集新文件时，从文件的开头开始读。
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

7. 日志格式、日志时间具体说明如下：

表 4-19 日志采集信息

名称	说明
日志格式	● 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 ● 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** ● 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 ● 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 ● 采集日志时间限制：系统时间的前后24小时内。
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 ● 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 ● 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond（999） hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

说明

时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。**如果没有特殊需求，建议使用单行日志-系统时间模式即可**。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考表4-20。

表 4-20 批量接入设置

类型	操作	说明
基本配置	接入类型	选择云主机 ECS-文本日志。
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考创建单个接入配置的操作步骤进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.8 ServiceStage 容器应用日志接入 LTS

云日志服务（Log Tank Service，简称LTS），用于收集来自容器CCE的日志数据，通过海量日志数据的分析与处理，可以将云服务和应用程序的可用性和性能最大化，为您提供实时、高效、安全的日志处理能力，帮助您快速高效地进行实时决策分析、设备运维管理、用户业务趋势分析等。

- 创建单个接入配置：创建ServiceStage-容器应用日志接入LTS的单个接入配置任务。

- **创建多个接入配置**：同时创建ServiceStage-容器应用日志接入LTS的多个接入配置任务。

说明

目前此功能仅支持白名单用户提交工单申请使用。详细操作请参考**提交工单**。

前提条件

- 已安装ICAgent并添加至主机组。
- 已创建ServiceStage应用。详细操作请参考**创建应用**。
- 已创建ServiceStage环境。详细操作请参考**创建环境**。
- 已创建ServiceStage组件。详细操作请参考**创建组件**。
- 开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

使用限制

- 支持容器引擎为Docker的CCE集群节点。
- 支持使用Containerd作为容器引擎的CCE集群节点（ICAgent 5.12.130及以上版本）。
- 容器内的日志目录如果已挂载到主机目录上，将无法通过配置容器文件路径方式采集到LTS，只能通过配置节点文件路径方式采集到LTS。
- Docker存储驱动限制：容器文件日志采集目前仅支持overlay2存储驱动，不支持devicemapper作为存储驱动的节点。查看存储驱动类型，请使用如下命令：

```
docker info | grep "Storage Driver"
```

创建单个接入配置的操作步骤

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，在接入向导页签或者在接入规则页签单击“接入日志”，单击“ServiceStage-容器应用日志”进行ServiceStage接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“ServiceStage-容器应用日志”进行ServiceStage接入配置。

步骤4 在选择日志流页面，设置如下参数。

1. 选择ServiceStage应用、ServiceStage环境。
2. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
3. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
4. 单击“下一步：检查依赖项”。

步骤5 检查依赖项。

系统自动检查以下内容检查项是否符合要求：

1. 存在自定义标识符为**k8s-log-应用ID**的主机组。
2. 存在名为**k8s-log-应用ID**的日志组。

如果以上内容检查项中，有任意一项不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。

说明

- **自动修复**：一键帮您完成以上内容检查项配置。
- **重新检查**：重新检查依赖项。

步骤6 选择主机组（可选）。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组。

   说明

   - 默认选择集群所在的主机组，您可以根据需要选择其他已创建的主机组。
   - 主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。
     - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
     - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。
2. 单击“下一步：采集配置”。

步骤7 采集配置。

设置具体的采集规则，具体可参考**采集配置**。

步骤8 结构化配置，具体请参考**结构化配置**。

说明

当所选日志流已配置结构化时，请谨慎执行删除操作。

若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

步骤9 索引配置，具体请参考**索引配置**。

步骤10 单击“提交”，完成ServiceStage接入。在接入规则页签，则会生成一条接入配置信息，支持以下操作：

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。
- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用ServiceStage接入完成日志接入时，采集配置的具体配置如下：

1. **基本配置**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置。
   - 容器标准输出：采集集群内指定容器日志，仅支持Stderr和Stdout的日志。

     说明

     - 被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。
     - 容器标准输出不能重复配置，即使跨日志组和日志流，也只能配置一次。
   - 容器文件路径：采集集群内指定容器内的文件日志。
   - 节点文件路径：采集集群内指定节点的文件。

     说明

     采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。
   - K8S事件：采集K8S集群内的事件日志。

     说明

     K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。

表 4-21 采集配置参数表

类型	参数配置
容器标准输出	采集容器标准输出（stdout）和采集容器标准错误（stderr）两者必须得有一个是开启状态。
容器文件路径	• 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** • 当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。 • 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。
节点文件路径	• 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。

类型	参数配置
K8S事件	无需设置参数。仅支持icagent 5.12.150 及以上版本。

3. ServiceStage匹配规则，选择对应组件。
4. 开启结构化解析配置，详细操作请参考**ICAgent结构化解析规则说明**。

   需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

   说明

   若已经配置了云端结构化解析，请先删除云端结构化解析后再配置ICAgent结构化解析。

图 4-6 ICAgent 结构化解析配置

5. 其他配置。

表 4-22 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启日志拆分按钮，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启日志拆分按钮，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。

名称	说明
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 ● 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 ● 全量采集：ICAgent采集新文件时，从文件的开头开始读。
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

6. 高级配置：日志格式、日志时间具体说明如下：

表 4-23 日志采集信息

名称	说明
日志格式	● 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 ● 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** ● 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 ● 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 ● 采集日志时间限制：系统时间的前后24小时内。

名称	说明
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 ● 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 ● 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond (999) hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考**表4-24**。

表 4-24 批量接入设置

类型	操作	说明
基本配置	接入类型	选择ServiceStage-容器应用日志。
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。

类型	操作	说明
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考**创建单个接入配置的操作步骤**进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.9 ServiceStage 云主机日志接入 LTS

云日志服务（Log Tank Service，简称LTS），用于收集来自云主机ECS的日志数据，通过海量日志数据的分析与处理，可以将云服务和应用程序的可用性和性能最大化，为您提供实时、高效、安全的日志处理能力，帮助您快速高效地进行实时决策分析、设备运维管理、用户业务趋势分析等。

- **创建单个接入配置**：创建ServiceStage-云主机日志接入LTS的单个接入配置任务。
- **创建多个接入配置**：同时创建ServiceStage-云主机日志接入LTS的多个接入配置任务。

说明

目前此功能仅支持白名单用户提交工单申请才能使用。详细操作请参考**提交工单**。

前提条件

- 已安装ICAgent并添加至主机组。
- 已创建ServiceStage应用。详细操作请参考**创建应用**。
- 已创建ServiceStage环境。详细操作请参考**创建环境**。
- 已创建ServiceStage组件。详细操作请参考**创建组件**。
- 开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

创建单个接入配置的操作步骤

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，在接入向导页签或者在接入规则页签单击“接入日志”，单击“ServiceStage-云主机日志”进行ServiceStage接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“ServiceStage-云主机日志”进行ServiceStage接入配置。

步骤4 在选择日志流页面，设置如下参数。

1. 选择ServiceStage应用、ServiceStage环境。
2. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
3. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
4. 单击“下一步：选择主机组（可选）”。

步骤5 选择主机组（可选）。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组。

   说明

   主机组可以为空，可以在接入配置设置完成后对主机组进行设置。
   - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
   - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。
2. 单击“下一步：采集配置”。

步骤6 采集配置。

设置具体的采集规则，具体可参考**采集配置**。

步骤7 云端结构化配置，具体请参考**设置云端结构化解析日志**。

说明

当所选日志流已配置结构化时，请谨慎执行删除操作。

若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

步骤8 索引配置，具体请参考**索引配置**。

步骤9 单击“提交”，完成ServiceStage接入。在接入规则页签，则会生成一条接入配置信息，支持以下操作：

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。

- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用ServiceStage接入完成日志接入时，采集配置的具体配置如下：

1. 采集配置名称：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。
   - **采集路径支持递归路径，**表示递归5层目录。**

     示例：采集路径配置为 /var/logs/**/a.log，日志匹配如下：

     ```
     /var/logs/1/a.log
     /var/logs/1/2/a.log
     /var/logs/1/2/3/a.log
     /var/logs/1/2/3/4/a.log
     /var/logs/1/2/3/4/5/a.log
     ```

     说明

     - 以上示例中的/1/2/3/4/5/，表示/var/logs目录中，往里递归的5个目录层级，在这5个目录层级中只要存在a.log，都能进行日志匹配。
     - 采集路径中只能出现一次**，不能出现两个及以上。正确示例：/var/logs/**/a.log；错误示例：/opt/test/**/log/**。
     - 采集路径中第一个层级不允许为**（避免误采集系统文件），错误示例：/**/test。
   - **采集路径支持模糊匹配，匹配目录或文件名中的任何字符**。

     说明

     如果配置了C:\windows\system32类似的日志采集路径，但无法采集日志，请尝试打开WAF物理防火墙后重新配置。

     - 示例1：采集路径配置为 /var/logs/*/a.log，表示/var/logs/目录下，任何一个目录中存在a.log，都能进行日志匹配，例如：

       /var/logs/1/a.log

       /var/logs/2/a.log

     - 示例2：采集路径配置为 /var/logs/service-*/a.log，日志匹配示例：

       /var/logs/service-1/a.log

       /var/logs/service-2/a.log

     - 示例3：采集路径配置为 /var/logs/service/a*.log，日志匹配示例：

       /var/logs/service/a1.log

       /var/logs/service/a2.log
   - **采集路径如果配置的是目录，示例：/var/logs/，则只采集目录下后缀为“.log”、“.trace”和“.out”的文件**。

     如果配置的是文件名，则直接采集对应文件，只支持内容是文本格式的文件。可以通过*file -i 文件名*命令，查询文件格式。

- 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。

  内置类型为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。

  自定义绕接规则为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log

说明

- 请注意您的敏感信息是否在收集范围内。
- 当主机选择“Windows主机”时，如需采集系统日志，需要在“采集配置”环节，开启“采集Windows事件日志”。
- windows事件日志采集不能重复配置，即相同主机下，即使跨日志组和日志流，也只能配置一次。
- LTS暂不支持采集PostgreSQL（数据库）实例的日志，目前只支持采集安装在ECS（主机）实例的日志。
- **日志采集路径不能重复配置**，即相同主机的同一个日志采集路径不能重复配置，否则可能会导致日志采集异常。
- 相同主机的同一个日志采集路径，如果在AOM进行了配置，则不能在LTS重复配置。
- 配置采集的文件最后修改时间和当前时间差如果已超过12小时，则不会采集。

3. 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。

   目录和文件名支持完全匹配，也支持模糊匹配，具体可参考**路径配置内容**进行设置。

   说明

   当设置的黑名单与配置的采集路径重复或者有重合时，优先过滤掉黑名单设置的文件。

4. 采集Windows事件日志：当选择Windows主机采集日志时，需要开启“采集Windows事件日志”，配置如下参数：

   表 4-25 采集 Windows 事件日志参数

名称	说明
日志类型	日志类型有系统、应用程序、安全和启动。
首次采集时间偏移量	如设置为7天，表示从采集开始时间前7天内的日志（7天前的日志被忽略），该时间仅在首次配置采集生效，确保不会重复采集。最大支持设置为7天。
事件等级	事件等级有information、warning、error、critical和verbose。根据Windows事件等级过滤采集。仅支持Windows Vista及以上的操作系统。

5. ServiceStage匹配规则，选择对应组件。
6. 开启结构化解析配置，详细操作请参考**ICAgent结构化解析规则说明**。

   需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

说明

若已经配置了云端结构化解析，请先删除云端结构化解析后再配置ICAgent结构化解析。

图 4-7 ICAgent 结构化解析配置

7. 其他配置。

表 4-26 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启日志拆分按钮，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启日志拆分按钮，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 ● 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 ● 全量采集：ICAgent采集新文件时，从文件的开头开始读。

名称	说明
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

8. 日志格式、日志时间具体说明如下：

表 4-27 日志采集信息

名称	说明
日志格式	● 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 ● 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** ● 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 ● 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 ● 采集日志时间限制：系统时间的前后24小时内。

名称	说明
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 ● 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 ● 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond (999) hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考**表4-28**。

表 4-28 批量接入设置

类型	操作	说明
基本配置	接入类型	选择ServiceStage-云主机日志。
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。

类型	操作	说明
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考**创建多个接入配置的操作步骤**进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.10 自建 K8s 应用日志接入 LTS

云日志服务（LTS）支持自建K8s-应用日志接入。

- **创建单个接入配置**：创建自建k8s - 应用日志接入LTS的单个接入配置任务。
- **创建多个接入配置**：同时创建自建k8s - 应用日志接入LTS的多个接入配置任务。

前提条件

- 请确保已在Kubernetes集群中执行安装Helm v3的命令。
- 请确保Kubernetes集群已配置kubectl。
- 开启“ICAgent诊断开关”用于查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控，请参考**设置ICAgent日志采集开关**。

创建单个接入配置的操作步骤

云日志服务接入方式选择自建k8s - 应用日志，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，在接入向导页签或者在接入规则页签单击“接入日志”，单击“自建k8s - 应用日志”进行自建K8s接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“云容器引擎 CCE-应用日志”进行CCE接入配置。

步骤4 选择日志流。

有两种采集方式：采集到集中日志流和采集到自定义日志流，您可以根据实际情况选择采集方式，**推荐您使用采集到集中日志流**。

采集到集中日志流

集中采集日志到一个固定的日志流。CCE集群默认的采集日志流分别为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、容器文件containerfile-{ClusterID}、K8S事件: event-{ClusterID}。日志流名称会根据ClusterID自动命名，例如：集群ID为Cluster01，则标准输出/错误日志流为stdout-Cluster01。

在一个CCE集群下可以创建的采集日志流为标准输出/错误stdout-{ClusterID}、节点文件hostfile-{ClusterID}、容器文件containerfile-{ClusterID}）和K8s事件event-{ClusterID}，如果某个日志组下，已创建某种采集日志流，则不会在其他日志组或当前日志组下再创建该日志流。

1. 选择采集方式“采集到集中日志流”。
2. 输入“集群名称”和“集群ID”。
3. 选择“所属日志组”。

   说明

   当无该日志组时，系统会提示：暂无该日志组，后续操作中，系统将会为您自动创建，创建完成后日志会集中采集到该日志组中。
4. 单击“下一步：检查依赖项”。

采集到自定义日志流

1. 选择采集方式“采集到自定义日志流”。
2. 输入“集群名称”和“集群ID”。
3. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
4. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
5. 单击“下一步：检查依赖项”。

图 4-8 采集到自定义日志流

步骤5 检查依赖项。

1. 系统自动检查以下三项是否符合要求：
   - 存在自定义标识为**k8s-log-集群ID**的主机组。
   - 存在名为**k8s-log-集群ID**的日志组。支持修改日志组的日志存储时间和备注。
   - 存在系统推荐的集中采集的日志流。支持修改日志流的日志存储时间和备注。当选择日志流为**采集到集中日志流**时，会进行该项内容检查。

   如果以上三项中，有任意一项不符合要求，需单击“自动修复”按钮进行修复，否则将无法进行下一步操作。

   说明

   - **自动修复**：一键帮您完成以上三项配置。
   - **重新检查**：重新检查依赖项。
   - 当选择日志流为**采集到自定义日志流**时，“存在名为**k8s-log-集群ID**的日志组”的检查项为可选项。您可以通过开启或关闭开关进行控制，确定是否进行该项检查。
2. 单击“下一步：安装日志采集组件”。

步骤6 安装日志采集组件。

在Kubernetes集群中，选择任意一台主机执行如下操作步骤：

1. 获取ICAgent安装包。
   - 获取ICAgent安装包（以界面上显示的为准）
     ```
     wget https://icagent-{regionId}.{obsDomainName}/ICAgent_linux/icagentK8s-5.5.1.2.tar.gz
     ```
   - 解压ICAgent安装包
     ```
     tar -xzvf icagentK8s-5.5.1.2.tar.gz
     ```
   - 进入目录
     ```
     cd icagentK8s
     ```


- 生成安装命令

  选择接入日志的**区域名**。

  选择接入日志的账号的**项目ID**。

  k8s集群所在区域，选择“区域内”。

2. 安装ICAgent。

   a. 复制ICAgent安装命令

   为了避免泄漏您的AK/SK，请勾选此处，执行关闭历史记录命令。

   图 4-9 安装 ICAgent

   生成安装命令如下：（*x.x.x.x*以界面上显示的实际IP为准）

   ```
   set +o history; bash icagent_log_install.sh 2a473356cca5487f8373be891bffc1cf test-xx123456 region0_id {input_your_ak} {input_your_sk} x.x.x.x podlb
   ```

   执行的命令需要填写AK/SK，有两种方式可选择：

   方式一：复制命令手动替换 {input_your_ak} 和 {input_your_sk}，填写值时不需要添加{}。

   方式二：直接执行复制的命令，系统会提示“Enter the AK”和“Enter the SK”，填写对应的AK/SK即可。

   b. 使用PuTTY等远程登录工具，以root用户登录待安装主机，执行复制到的命令。

   当显示”ICAgent install success”时，表示安装成功，安装成功后，在左侧导航栏中选择“主机管理”，查看ICAgent状态。

3. 单击“确认安装完毕”。

步骤7 选择主机组（可选）。

1. 在主机组列表中选择一个或多个需要采集日志的主机组，若没有所需的主机组，单击列表左上方“新建”，在弹出的新建主机组页面创建新的主机组，具体可参考**创建主机组（自定义标识）**。

   说明

   - 默认选择集群所在的主机组，您可以根据需要选择其他已创建的主机组。
   - 主机组可以为空，但是会导致采集配置不生效，建议第一次接入时选择主机组。若不选择，可以在接入配置设置完成后对主机组进行设置。
     - 在“主机管理 > 主机组”页面对主机组和接入配置进行关联。
     - 在接入规则页面，单击操作列的编辑，进入接入配置页面对主机组和接入配置进行关联。

2. 单击“下一步：采集配置”。

步骤8 采集配置。

1. 设置具体的采集规则，具体可参考**采集配置**。

2. 单击“下一步：结构化配置”。

步骤9 结构化配置（可选项）。

1. 单击“跳过”或进行结构化配置，具体请参考**设置云端结构化解析日志**。

   说明

   当所选日志流已配置结构化时，请谨慎执行删除操作。

   若配置ICAgent结构化解析配置功能后，则不需要配置云端结构化解析了。详细请参考**ICAgent结构化解析规则说明**。

2. 单击“下一步：索引配置”。

步骤10 索引配置（可选项）。

单击“跳过并提交”或进行索引配置，具体请参考**设置LTS日志索引配置**。

步骤11 配置完成后单击“提交”，完成接入。在接入规则页签，则会生成一条接入配置信息。

- 单击接入配置名称可进入详情页面，查看该接入配置详细信息。
- 单击接入配置操作列的“编辑”重新修改接入配置信息。
- 单击接入配置操作列的“标签管理”即可添加标签。
- 单击接入配置操作列的“复制”复制一条新的接入配置信息。
- 单击接入配置操作列的“删除”即可删除接入配置信息。
- 单击接入配置操作列的“采集诊断”，可查看ICAgent异常监控、ICAgent整体状态和ICAgent采集监控。

----结束

采集配置

在使用自建K8s接入完成日志接入时，采集配置的具体配置如下：

1. **基本配置**：自定义采集配置名称，长度范围为1到64个字符，只支持输入英文、数字、中文、中划线、下划线以及小数点，且不能以小数点、下划线开头或以小数点结尾。
2. **数据源配置**：选择数据源类型，进行对应的数据源配置。
   - 容器标准输出：采集集群内指定容器日志，仅支持Stderr和Stdout的日志。

     说明

     - 被匹配上的容器的标准输出会采集到指定的日志流，原先采集到的AOM的标准输出会停止采集。
     - 容器标准输出不能重复配置，即使跨日志组和日志流，也只能配置一次。
   - 容器文件路径：采集集群内指定容器内的文件路径日志。
   - 节点文件路径：采集集群内指定节点路径的文件。

     说明

     采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。
   - K8S事件：采集K8S集群内的事件日志。

说明

K8S事件不能重复配置，即一个K8S集群的K8S事件，只能配置接入到一个日志流。

表 4-29 采集配置参数表

类型	参数配置
容器标准输出	采集容器标准输出（stdout）和采集容器标准错误（stderr）。两者必须得有一个是开启状态。 开启后采集容器标准错误（stderr），可以选择采集目前路径：将标准输出和标准错误采集到不同的文件（stdout.log和stderr.log）、将标准输出和标准错误采集到同一个文件（stdout.log）。
容器文件路径	• 路径配置：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** • 当CCE集群的工作负载中，已配置容器的挂载路径时，此时路径配置里添加的路径将无效。须将CCE集群页面中的挂载路径删除后，该配置才有效。 • 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。 **内置类型**为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。 **自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。

类型	参数配置
节点文件路径	• **路径配置**：添加您需要收集的日志路径，LTS将按照配置的路径进行日志采集。 **说明** 采集路径不能重复配置，即同一个主机下的同一路径，即使跨日志组和日志流，也只能配置一次。 • 添加自定义绕接规则，ICAgent目前是通过文件名规则来判断是否为绕接文件，如果您的绕接规则不符合内置类型时，可以通过单击“添加自定义绕接规则”来进行匹配，避免重复采集和绕接时的日志丢失。 **内置类型**为{basename}{连接符}{绕接标识}.后缀，{basename}.{后缀}{连接符}{绕接标识}。其中连接符为-._绕接标识为非字母符号，后缀为字母。 **自定义绕接规则**为{basename}+绕接文件的特征正则表达式组成匹配规则。例如您的日志文件名称为/opt/test.out.log，绕接后的文件名为test.2024-01-01.0.out.log，test.2024-01-01.1.out.log，因此在路径配置时，采集路径为/opt/*.log，绕接规则为{basename}\.[-0-9\.].out.log • 设置采集黑名单：LTS支持对日志进行过滤采集，即通过设置黑名单，在采集时过滤指定的目录或文件。指定按目录过滤，可过滤掉该目录下的所有文件。
K8S事件	无需设置参数。仅支持icagent 5.12.150 及以上版本。

3. K8s匹配规则：当数据源类型选择容器标准输出和容器文件路径时，设置K8s匹配规则，非必选项。

   说明

   填写正则匹配规则后，单击校验按钮，支持校验确保正则表达式的正确性。

   表 4-30 K8s 匹配规则

参数名称	参数说明
K8s Namespace正则匹配	通过Namespace名称指定采集的容器，支持正则匹配。 **说明** 采集名称符合正则规则的Namespace的日志，为空时采集所有Namespace的日志。
K8s Pod正则匹配	通过Pod名称指定待采集的容器，支持正则匹配。 **说明** 采集名称符合正则规则的Pod的日志，为空时采集所有Pod的日志。
K8s容器名称正则匹配	通过容器名称指定待采集的容器（Kubernetes容器名称是定义在spec.containers中），支持正则匹配。 **说明** 采集名称符合正则规则的容器的日志，为空时采集所有容器的日志。

参数名称	参数说明
容器Label白名单	通过容器Label白名单指定待采集的容器。如果您要设置容器Label白名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则容器 Label中包含LabelKey的容器都匹配；若LabelValue不为空，则容器 Label中包含LabelKey=LabelValue的容器才匹配；LabelKey需要全匹配，LabelValue支持正则匹配；多个白名单之间为或关系，即只要容器 Label满足任一白名单即可被匹配。
容器Label黑名单	通过容器Label黑名单排除不采集的容器。如果您要设置容器Label黑名单，那么LabelKey必填，LabelValue可选填。 **说明** 若LabelValue为空，则容器 Label中包含LabelKey的容器都被排除；若LabelValue不为空，则容器 Label中包含LabelKey=LabelValue的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配；多个黑名单之间为或关系，即只要容器 Label满足任一黑名单即可被排除。
容器Label日志标签	设置容器Label日志标签后，日志服务将在日志中新增容器Label相关字段。 **说明** 设置容器 Label日志标签后，lts将在日志中新增相关字段。例如设置LabelKey为app，设置LabelValue为app_alias，当容器中包含app=lts时，将在日志中添加的内容{app_alias：lts}。
环境变量白名单	用于指定待采集的容器。如果您要设置环境变量白名单，那么Label Key必填，Label Value可选填。 **说明** 如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都匹配；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才被匹配；LabelKey需要全匹配，LabelValue支持正则匹配；多个白名单之间为或关系，即只要容器的环境变量满足任一键值对即可被匹配。
环境变量黑名单	用于排除不采集的容器。如果您要设置环境变量黑名单，那么Label Key必填，Label Value可选填。 **说明** 如果环境变量Value为空，则容器环境变量中包含环境变量Key的容器都将被排除；如果环境变量Value不为空，则容器环境变量中包含环境变量Key=环境变量Value的容器才会被排除；LabelKey需要全匹配，LabelValue支持正则匹配；多个黑名单之间为或关系，即只要容器的环境变量满足任一键值对即可被排除。
环境变量日志标签	设置环境变量日志标签后，日志服务将在日志中新增环境变量相关字段。 **说明** 设置环境变量日志标签后，lts将在日志中新增相关字段，例如设置环境变量Key为app，设置环境变量Value为app_alias，当容器中包含环境变量app=lts时，将在日志中添加的内容为{app_alias：lts}。

4. 开启结构化解析配置，详细操作请参考**ICAgent结构化解析规则说明**。

   需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

说明

若已经配置了云端结构化解析，请先删除云端结构化解析后再配置ICAgent结构化解析。

图 4-10 ICAgent 结构化解析配置

5. 其他配置。

表 4-31 其他配置

名称	说明
最大目录深度	最大目录深度为20层。 采集路径支持使用**配置多层路径模糊匹配，该配置项限制最大目录深度。例如您的日志路径为/var/logs/department/app/a.log，采集路径配置为：/var/logs/**/a.log，当配置为1时日志不会被采集，配置>=2时日志会被采集。
日志拆分	云日志服务支持对日志进行拆分。 当日志大小超过500KB时，开启日志拆分按钮，则单行日志会被拆分为多行采集。支持设置日志拆分大小，最大为1024KB。例如：日志大小为600KB，被拆分为2行日志采集，第一行500KB，第二行100KB。 当日志大小超过500KB时，未开启日志拆分按钮，则单条日志大小限制不超过500KB，超过限制部分会被截断丢弃。
采集二进制文件	云日志服务支持采集二进制文件。 您可以通过命令（*file -i 文件名*）查看文件类型，如果包含charset=binary，那么该日志文件就是二进制文件。 当日志的文件类型为二进制时，开启采集二进制文件按钮，则对接入的二进制文件日志进行采集，但仅支持UTF8编码的字符串，非UFT8编码的字符在LTS控制台页面会显示乱码。 当日志的文件类型为二进制时，未开启采集二进制文件按钮，则对接入的二进制文件日志停止采集，开启后即可进行采集。
日志文件编码	日志文件编码为UTF-8。
采集策略	采集策略支持增量或全量。 • 增量采集：ICAgent采集新文件时，从文件的末尾开始读。 • 全量采集：ICAgent采集新文件时，从文件的开头开始读。

名称	说明
自定义元数据	默认开启自定义元数据，ICAgent会根据您选择的内置字段和自定义键值对上传LTS。否则，ICAgent会使用默认配置。
系统内置字段	需要开启自定义元数据后，才能设置系统内置字段。
自定义键值对	需要开启自定义元数据后，才能设置自定义键值对。单击添加，输入键值key和键值Value。

6. 日志格式、日志时间具体说明如下：（首次创建配置接入时不显示以下参数，配置完成后，编辑配置信息时才会显示以下参数。）

   说明

   不再推荐使用以下功能，建议使用**结构化解析配置**。

   如果您配置了ICAgent的多行全文或者多行完全正则，此处的多行日志配置会失效。

   表 4-32 日志采集信息

名称	说明
日志格式	• 单行日志：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。 • 多行日志：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
日志时间	系统时间：表示系统当前时间，默认为日志采集时间，每条日志的行首显示日志的采集时间。 **说明** • 日志采集时间：ICAgent采集日志，并且发送到云日志服务的时间。 • 日志打印时间：系统产生并打印日志的时间。ICAgent采集日志并发送日志到云日志平台的频率为1秒钟。 • 采集日志时间限制：系统时间的前后24小时内。

名称	说明
	时间通配符：用日志打印时间来标识一条日志数据，通过时间通配符来匹配日志，每条日志的行首显示日志的打印时间。 ● 如果日志中的时间格式为：2019-01-01 23:59:59.011，时间通配符应该填写为：YYYY-MM-DD hh:mm:ss.SSS。 ● 如果日志中的时间格式为：19-1-1 23:59:59.011，时间通配符应该填写为：YY-M-D hh:mm:ss.SSS。 **说明** 如果日志中不存在年份信息，则云日志会自动补齐年份数据为当前年份数据。 填写示例： YY - year (19) YYYY - year (2019) M - month (1) MM - month (01) D - day (1) DD - day (01) hh - hours (23) mm - minutes (59) ss - seconds (59) SSS - millisecond (999) hpm - hours (03PM) h:mmpm - hours:minutes (03:04PM) h:mm:sspm - hours:minutes:seconds (03:04:05PM) hh:mm:ss ZZZZ (16:05:06 +0100) hh:mm:ss ZZZ (16:05:06 CET) hh:mm:ss ZZ (16:05:06 +01:00)
分行模式	日志格式选择多行日志时，需要选择分行模式，分行模式选择“日志时间”时，是以时间通配符来划分多行日志；当选择“正则模式”时，则以正则表达式划分多行日志。
正则表达式	此配置是用来标识一条日志数据的正则表达式。日志格式选择“多行日志”格式后且“分行模式”已选择“正则模式”后需要设置。

说明

时间通配和正则表达式均是从每行日志的开头进行严格匹配，如果匹配不上，则会默认使用系统时间上报，这样可能会和文件内容中的时间不一致。**如果没有特殊需求，建议使用单行日志-系统时间模式即可**。

创建多个接入配置的操作步骤

在接入规则页签，支持创建批量接入的任务。

步骤1 支持批量创建接入，单击“批量接入”，进入配置详情页面，请参考**表4-33**。

表 4-33 批量接入设置

类型	操作	说明
基本配置	接入类型	选择自建k8s - 应用日志。

类型	操作	说明
	接入配置数量	在输入框填写接入配置数量，单击“添加接入配置”。 在接入配置下方默认已有1个接入配置，最多支持再添加99个数量，因此支持同时添加100个接入配置。
接入配置	接入列表	1. 左侧显示接入配置的信息，最多支持添加99个配置。 2. 右侧显示配置接入的内容，详细请参考**创建单个接入配置的操作步骤**进行设置。 3. 一个接入配置设置完成后，单击“应用于其他接入规则”即可将该接入配置复制到其他接入配置。

步骤2 单击参数检查，检查成功后，单击“提交”，批量接入设置完成。

步骤3 例如添加了4个接入配置，批量创建成功后，在接入规则页签下方，就会显示4条接入配置数量。

步骤4 （可选）支持对接入配置任务进行以下操作：

- 勾选多个已创建成功的接入配置，单击“批量编辑”进入配置详情页面，通过选择不同接入类型，修改对应的接入配置信息。
- 勾选多个已创建成功的接入配置，单击开启或关闭按钮。接入配置状态关闭后不会继续采集日志。
- 勾选多个已创建成功的接入配置，单击删除按钮即可批量删除接入配置。

----结束

4.2.11 ICAgent 结构化解析规则说明

云日志服务LTS支持通过ICAgent采集方式进行日志上报。在创建日志接入时设置采集配置策略，例如解析规则、白名单规则、黑名单规则、上传原始日志等参数，实现定制化的采集策略。ICAgent采集配置定义了如何在服务器上采集同类日志并解析、发送到指定的日志流上。

功能优势

- 基于日志文件，无侵入式采集日志。您无需修改应用程序代码，且采集日志不会影响您的应用程序运行。
- 稳定处理日志采集过程中的各种异常。当遇到网络异常、服务端异常等问题时会采用主动重试、本地缓存数据等措施保障数据安全。
- 基于日志服务的集中管理能力。安装ICAgent后，只需要在日志服务上配置主机组、ICAgent采集配置等信息即可。
- 完善的自我保护机制。为保证运行在服务器上的ICAgent，不会明显影响您服务器上其他服务的性能，ICAgent在CPU、内存及网络使用方面都做了严格的限制和保护机制。

日志接入前，您可以提前了解ICAgent采集的结构化解析规则，方便您快速操作。需要ICAgent 5.12.147及以上版本，其优点是成本更低，支持组合解析，一个日志流的每个采集配置可以配置不同的结构化解析规则。

支持以下日志结构化解析规则：

- **单行-全文日志**：采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。
- **多行-全文日志**：采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。
- **JSON**：适用JSON格式的日志，通过提取JSON字段将其拆分为键值对。
- **分隔符**：使用分隔符（例如：逗号、空格或字符）提取字段。
- **单行-完全正则**：适用任意格式的单行日志，使用正则表达式提取字段。填写正则匹配规则后，单击验证按钮，支持校验确保正则表达式的正确性。
- **多行-完全正则**：适用任意格式的多行日志，使用正则表达式提取字段。首行正则表达式支持自动生成和手动输入，填写正则匹配规则后，单击验证按钮，支持校验确保正则表达式的正确性。
- **组合解析**：适用于多格式嵌套的日志（例如：分隔符+JSON）。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

单行-全文日志

采集的日志文件中，如果您希望每一行日志在LTS界面中都显示为一条单独的日志数据，则选择单行日志。

1. 选择单行-全文日志。
2. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
3. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
4. 开启日志过滤才需要设置，添加黑名单规则。

您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*

多行-全文日志

采集的日志中包含像java异常的日志，如果您希望多行异常的日志显示为一条日志，正常的日志则每一行都显示为一条单独的日志数据，则选择多行日志，方便您查看日志并且定位问题。

1. 选择多行-全文日志。
2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，手动输入或自动生成首行正则表达式。
   - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
   - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

3. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
4. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
5. 开启日志过滤才需要设置，添加黑名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*

JSON

适用JSON格式的日志，通过提取JSON字段将其拆分为键值对。

1. 选择JSON格式。
2. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
3. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
4. 开启日志过滤才需要设置，添加黑名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
5. 上传原始日志。

   打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
6. 上传解析失败日志。

   打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。
7. **自定义日志时间**。

   开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。
8. json解析层数。增加json解析层数配置，取值范围为1~4，只能整数，默认值为1。

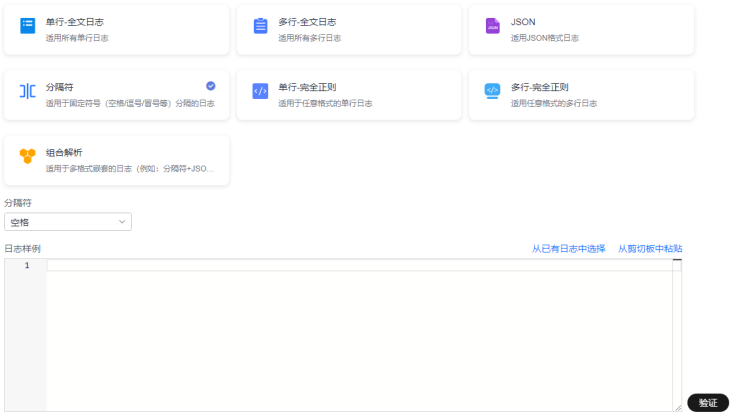
   将json格式日志的字段展开，例如原始日志为{"key1":{"key2":"value"}}，解析1层日志为：{"key1":{"key2":"value"}}，解析2层日志为：{"key1.key2":"value"}。

分隔符

使用分隔符（例如：逗号、空格或字符）提取字段。

1. 选择分隔符。
2. 根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。
3. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，单击“验证”，在提取结果下方查看结果。
   - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
   - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

图 4-11 分隔符



4. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
5. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
6. 开启日志过滤才需要设置，添加黑名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
7. 上传原始日志。

   打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
8. 上传解析失败日志。

   打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。
9. **自定义日志时间**。

   开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

单行-完全正则

适用任意格式的单行日志，使用正则表达式提取字段。

1. 选择单行-完全正则。
2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，在提取正则表达式下方的输入框输入要提取日志的正则表达式，单击“验证”，在提取结果下方查看结果。
   - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
   - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

3. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
4. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
5. 开启日志过滤才需要设置，添加黑名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
6. 上传原始日志。

   打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。
7. 上传解析失败日志。

   打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。
8. **自定义日志时间**。

   开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

多行-完全正则

适用任意格式的多行日志，使用正则表达式提取字段。

1. 选择多行-完全正则。
2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，手动输入或自动生成首行正则表达式，在提取正则表达式下方的输入框输入要提取日志的正则表达式，单击“验证”，在提取结果下方查看结果。
   - 从已有日志中选择：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
   - 从剪切板中粘贴：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。
3. 日志过滤默认关闭，可根据需要打开日志过滤，进行添加白名单规则或黑名单规则，白名单规则或黑名单规则添加上限为20个。
4. 开启日志过滤才需要设置，添加白名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为命中规则，即匹配上正则表达式的日志才会被采集上报。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过滤规则之间关系是“或”逻辑，例如采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*
5. 开启日志过滤才需要设置，添加黑名单规则。

   您可以添加过滤规则筛选出有价值的日志数据，过滤规则为正则表达式，应用到指定Key的Value，所创建的过滤规则为丢弃规则，即匹配上正则表达式的日志会被丢弃。单行/多行全文模式下，默认使用content作为全文的键{key}名；多条过

滤规则之间关系是“或”逻辑，例如不采集日志源文中包含hello的日志，可配置采集规则为：.*hello.*

6. 上传原始日志。

   打开上传原始日志开关后，原始日志将作为content字段的值上传到日志服务。

7. 上传解析失败日志。

   打开上传解析失败日志开关后，原始日志将作为_content_parse_fail_字段的值上传到日志服务。

8. **自定义日志时间**。

   开启后可指定某一字段作为日志时间，或关闭此项使用日志被采集时间作为日志时间。

组合解析

适用于多格式嵌套的日志（例如：分隔符+JSON），根据语法自定义配置解析规则。

1. 选择组合解析。
2. 从“从已有日志中选择”或“从剪切板中粘贴”选择日志样例，在插件配置下方输入配置内容。
3. 您可以根据日志内容参考以下插件语法自定义设置：
   - processor_regex

     表 4-34 正则提取

参数	类型	说明
source_key	string	原始字段名。
regex	string	正则表达式()中为提取字段。
keys	string	为提取的内容指定字段名。
keep_source	boolean	是否保留原始字段。
keep_source_if_parse	boolean	解析错误是否保留原始字段。

   - processor_split_string

     表 4-35 分隔符解析

参数	类型	说明
source_key	string	原始字段名。
split_sep	string	分隔符字符串。
keys	string	为提取的内容指定字段名。
keep_source	boolean	被解析后的日志中是否保留原始字段。

参数	类型	说明
split_type	char/special_char/string	分隔类型，支持char-单字符、special_char-不可见字符、string-字符串。
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。

- processor_split_key_value

表 4-36 键值对分割

参数	类型	说明
source_key	string	原始字段名。
delimiter	string	键值对之间的分隔符，默认值为制表符\t。
separator	string	单个键值对中键与值之间的分隔符，默认值为半角冒号（:）。
keep_source	boolean	被解析后的日志中是否保留原始字段。

- processor_add_fields

表 4-37 添加字段

参数	类型	说明
fields	json/object	待添加的字段名和字段值。键值对格式，支持添加多个。

- processor_drop

表 4-38 丢弃字段

参数	类型	说明
drop_keys	string	丢弃的字段列表。

- processor_rename

表 4-39 重命名字段

参数	类型	说明
source_keys	string	待重命名的原始字段。
destkeys	string	重命名后的字段。

- processor_json

  表 4-40 json 展开提取

参数	类型	说明
source_key	string	原始字段名。
keep_source	string	被解析后的日志中是否保留原始字段。
expand_depth	int	json展开的深度。默认值为0，表示不限制。1表示当前层级，以此类推。
expand_connector	string	json展开时的连接符，默认值为下划线（_）。
prefix	string	json展开时，对字段名附加的前缀。
keep_source_if_parse_error	boolean	解析错误是否保留原始字段。

- processor_filter_regex

  表 4-41 过滤器

参数	类型	说明
include	json/object	key为日志字段，value为匹配的正则表达式。
exclude	json/object	key为日志字段，value为匹配的正则表达式。

- processor_gotime

  表 4-42 提取时间

参数	类型	说明
source_key	string	原始字段名。
source_format	string	原始时间的格式。
source_location	int	原始时间的时区。参数值为空时，表示logtail所在主机或容器的时区。
dest_key	string	解析后的目标字段。
dest_format	string	解析后的时间格式。
dest_location	int	解析后的时区。参数值为空时，表示本机时区。

参数	类型	说明
set_time	boolean	是否将解析后的时间设置为日志时间。
keep_source	boolean	被解析后的日志中是否保留原始字段。

4. 参考示例：

```
[
    {
        "type": "processor_regex",
        "detail": {
            "source_key": "content",
            "regex": "*",
            "keys": [
                "key1",
                "key2"
            ],
            "multi_line_regex": "*",
            "keep_source": true,
            "keep_source_if_parse_error": true
        }
    },
    {
        "type": "processor_split_string",
        "detail": {
            "split_sep": ".",
            "split_type": ".",
            "split_keys": [
                "key1",
                "key2"
            ],
            "source_key": "context",
            "keep_source": true,
            "keep_source_if_parse_error": true
        }
    },
    {
        "type": "processor_add_fields",
        "detail": {
            "fields": [
                {
                    "key1": "value1"
                },
                {
                    "key2": "value2"
                }
            ]
        }
    },
    {
        "type": "processor_drop",
        "detail": {
            "drop_keys": [
                "key1",
                "key2"
            ]
        }
    },
    {
        "type": "processor_rename",
        "detail": {
            "source_key": [
                "skey1",
                "skey2"
```



```
            ],
            "dest_keys": [
               "dkey1",
               "dkey2"
            ]
         }
      },
      {
         "type": "processor_json",
         "detail": {
            "source_key": "context",
            "expand_depth": 4,
            "expand_connector": "_",
            "prefix": "prefix",
            "keep_source": true,
            "keep_source_if_parse_error": true
         }
      },
      {
         "type": "processor_gotime",
         "detail": {
            "source_key": "skey",
            "source_format": "ydm",
            "source_location": 8,
            "dest_key": "dkey",
            "dest_format": "ydm",
            "dest_ocation": 8,
            "set_time": true,
            "keep_source": true,
            "keep_source_if_parse_error": true
         }
      },
      {
         "type": "processor_filter_regex",
         "detail": {
            "include": {
               "ikey1": "*",
               "ikey2": "*"
            },
            "exclude": {
               "ekey1": "*",
               "ekey1": "*"
            }
         }
      }
   ]
```

自定义日志时间

开启自定义日志时间开关  ，参考**#lts_07_0072/zh-cn_topic_0000001679121953_table788424883516**设置参数。

说明

- 若时间格式填写错误或指定字段不存在，将使用日志被采集时间作为日志时间。
- 对结构化解析进行字段名称修改、字段删除、字段类型修改等操作，都需要重新校验时间字段。

参数配置表

参数	说明	示例
时间字段key名称	已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。	test

参数	说明	示例
字段value	已提取的字段value，选择字段key后，将自动填充。 **说明** 配置的字段value必须是当前时间前后24小时内的时间。	2023-07-19 12:12:00
时间格式	请参考**常见日志时间格式**。	yyyy-MM-dd HH:mm:ss
操作	单击  校验图标，提示“时间格式和字段value匹配成功”则表示校验成功。	-

常见日志时间格式

支持的常见日志时间格式如下表所示。

说明

默认情况下，日志服务中的日志时间戳精确到秒，所以时间格式只需配置到秒，无需配置毫秒、微秒等信息。

表 4-43 时间格式

时间格式	说明	示例
EEE	星期的缩写。	Fri
EEEE	星期的全称。	Friday
MMM	月份的缩写。	Jan
MMMM	月份的全称。	January
dd	每月第几天，十进制，范围为01~31。	07, 31
HH	小时，24小时制。	22
hh	小时，12小时制。	11
MM	月份，十进制，范围为01~12。	08
mm	分钟，十进制，范围为00~59。	59
a	AM或PM。	AM、PM
hh:mm:ss a	12小时制的时间组合。	11:59:59 AM
HH:mm	小时和分钟组合。	23:59
ss	秒数，十进制，范围为00~59。	59
yy	年份，十进制，不带世纪，范围为00~99。	04、98

时间格式	说明	示例
yyyy	年份，十进制。	2004、1998
d	每月第几天，十进制，范围为1~31。 如果是个位数字，前面需要加空格。	7、31
DDD	一年中的天数，十进制，范围为001~366。	365
u	星期几，十进制，范围为1~7，1表示周一。	2
w	每年的第几周，星期天是一周的开始，范围为00~53。	23
w	每年的第几周，星期一是一周的开始，范围为01~53。 如果一月份刚开始的一周>=4天，则认为是第1周，否则认为下一个星期是第1周。	24
U	星期几，十进制，范围为0~6，0代表周日。	5
EEE MMM dd HH:mm:ss yyyy	标准的日期和时间。	Tue Nov 20 14:12:58 2020
EEE MMM dd yyyy	标准的日期，不带时间。	Tue Nov 20 2020
HH:mm:ss	标准的时间，不带日期。	11:59:59
%s	Unix时间戳。	147618725

4.3 使用云服务接入 LTS

4.3.1 云服务接入 LTS 概述

云日志服务（LTS）支持采集计算、存储、安全、数据库等多种华为云服务的日志数据，您可以使用LTS对云服务日志进行关键词搜索、运营数据统计分析、运行状况监控告警等多种操作。当前LTS支持采集的云服务日志如下表所示：

表 4-44 云服务接入

序号	云服务简称	云服务名称	日志类型（结构化模板名称）	文档链接	仪表盘
1	AOM	应用运维管理	AOM接入的云服务日志 **（不推荐该方式）**	**应用运维管理AOM接入LTS**	-
2	APIG	API网关	网关访问日志（APIG）	**API网关APIG接入LTS**	• APIG监控中心 • APIG访问中心 • APIG秒级监控
4	BMS	裸金属服务器	全量日志（ICAgent采集文本日志）	**裸金属服务BMS文本日志接入LTS**	-
5	CBH	云堡垒机	操作日志	**云堡垒机CBH接入LTS**	-
6	CCE	云容器引擎	• 用户应用日志 • CCE管理面日志	**云容器引擎CCE应用日志接入LTS**	-
8	CFW	云防火墙	• 攻击日志（CFW攻击日志） • 访问日志（CFW访问控制日志） • 流量日志（CFW流量日志）	**云防火墙CFW接入LTS**	-
9	CTS	云审计服务	云服务管理面操作日志（CTS）	**云审计服务CTS接入LTS**	-
11	DDS	文档数据库服务	• 审计日志（DDS审计日志） • 错误日志（DDS错误日志） • 慢日志（DDS慢日志）	**文档数据库服务DDS接入LTS**	DDS审计日志中心

序号	云服务简称	云服务名称	日志类型（结构化模板名称）	文档链接	仪表盘
13	DMS	分布式消息服务Kafka版	DMS重平衡日志	分布式消息服务Kafka版接入LTS	-
14	DRS	数据复制服务	访问日志	数据复制服务DRS接入LTS	-
15	DWS	数据仓库服务	• CN节点日志 • DN节点日志 • 操作系统messages日志 • 审计日志	数据仓库服务GaussDB(DWS)接入LTS	-
16	ECS	弹性云服务器	全量日志（ICAgent采集文本日志）	云主机ECS文本日志接入LTS	-
17	ELB	弹性负载均衡	7层访问日志（ELB）	弹性负载均衡ELB接入LTS	• ELB监控中心 • ELB访问中心 • ELB秒级监控
18	ER	企业路由器	全量日志（ER企业路由器）	企业路由器ER接入LTS	-
19	FunctionGraph	函数工作流	函数执行日志	函数工作流FunctionGraph接入LTS	-
20	GaussDB	云数据库GaussDB	审计日志（GAUSSV5审计日志）	云数据库GaussDB接入LTS	-
21	GES	图引擎服务	审计日志	图引擎服务GES接入LTS	-
22	GaussDB for MySQL	云数据库GaussDB for MySQL	• 错误日志（GAUSSDB_MYSQL错误日志） • 慢日志（GAUSSDB_MYSQL慢日志）	云数据库GaussDB(for MySQL)接入LTS	-

序号	云服务简称	云服务名称	日志类型（结构化模板名称）	文档链接	仪表盘
23	GeminiDB Redis	云数据库GeminiDB Redis	慢日志（GeminiDB Redis慢日志）	云数据库GeminiDB接入LTS	-
27	IoTDA	设备接入	设备运行日志	设备接入IoTDA接入LTS	-
28	ModelArts	ModelArts	运行日志	AI开发平台ModelArts接入LTS	-
29	MRS	MapReduce服务	全量日志（ICAgent采集文本日志）	MapReduce服务MRS接入LTS	-
31	RDS for MySQL	云数据库 RDS for MySQL	• 错误日志（MYSQL错误日志） • 慢日志（MYSQL慢日志）	云数据库RDS for MySQL接入LTS	-
32	RDS for PostgreSQL	云数据库 RDS for PostgreSQL	• 错误日志（POSTGRESQL错误日志） • 慢日志（POSTGRESQL慢日志）	云数据库RDS for PostgreSQL接入LTS	-
33	RDS for SQLServer	云数据库RDS for SQL Server	错误日志（SQLSERVER错误日志）	云数据库RDS for SQLServer接入LTS	-
34	ROMA Connect	应用与数据集成平台	网关访问日志	应用与数据集成平台ROMA Connect接入LTS	-
36	SMN	消息通知服务	消息传输日志（SMN）	消息通知服务SMN接入LTS	-
37	SecMaster	安全云脑	全量日志	安全云脑secMaster接入LTS	-
38	ServiceStage	应用管理与运维平台	ServiceStage容器应用日志	ServiceStage容器应用日志接入LTS	-

序号	云服务简称	云服务名称	日志类型（结构化模板名称）	文档链接	仪表盘
39	VPC	虚拟私有云	VPC流日志（VPC）	虚拟私有云VPC接入LTS	VPC流日志
40	WAF	Web应用防火墙	• 攻击日志 • 访问日志	Web应用防火墙WAF接入LTS	-

4.3.2 应用运维管理 AOM 接入 LTS

LTS支持应用运维管理（AOM）日志接入，具体接入方法请参见接入LTS。

4.3.3 API 网关 APIG 接入 LTS

云日志服务支持API网关（API Gateway）日志接入。

前提条件

创建并使用华为云APIG实例。

设置 APIG 接入

云日志服务接入方式为API网关 APIG时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，单击“API网关 APIG”进行APIG接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步”： APIG配置。

步骤4 APIG配置。

单击“前往APIG配置”。

1. 在APIG控制台，选择API监控分析下的“日志分析”。
2. 单击“配置访问日志”链接。
3. 在弹出的配置访问日志页面中，选择对应的日志组和日志流。

说明

APIG里具体的操作请参见日志分析。

步骤5 单击下一步：日志流配置。

表 4-45 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启该按钮，自动对日志流进行结构化配置和索引配置。日志流结构化配置为APIG系统模板；索引配置为所有APIG解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对APIG日志进行SQL分析，并提供可视化图表。
自动为日志流添加标签：log_type=apig_layer_access	开启该按钮，自动为日志流添加标签（log_type=apig_access）后，APIG仪表盘模板才能匹配该日志流。
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建APIG仪表盘。

步骤6 完成。

----结束

结构化模板日志详情

- 结构化模板示例

表 4-46 结构化模板示例

模板名称	示例日志
APIG	100.125.7.59 f57f6523b675504a23887d0f5c1c8ef3 f5ea2360a2fa443cac236b76f4052ad6 - - [27/Jan/2022:15:56:44 +0800] 0.113 GET http://c965898968af48248ec7fac4ec0666f4.apic.cn-north-4.huaweicloudapis.com /api/echo HTTP/1.1 200 1443 408 "APIGatewayDebugClient/1.0" "-" "100.125.2.39:443" /v2/x/fgs/functions/urn:fss:cn-north-4:106506b9a92342df9a5025fc12351cfc:function:default:apigDemo_1640743997661:latest/invocations "200" "0.010" "0.083" "0.083" cn-north-4 0.083 0 - - - 0.03000020980835 - - "-" 486 HttpEchoDemo - - - "-" "-" "-" "-" "-" "-" "-" "-" "-" "-" "remote" "0.52" "http" "id" "-" "urn:fss:cn-north-7:6650ff66e4524699bea4c96ff2f268fe:function:default:test123hxy"

- 结构化字段详情

表 4-47 结构化字段

字段	示例	描述	类型
my_remote_addr	100.125.7.59	用户的客户端IP地址	string
request_id	f57f6523b675504a23887d0f5c1c8ef3	请求ID	string

字段	示例	描述	类型
api_id	f5ea2360a2fa443cac236b76f4052ad6	API ID	string
user_id	-	用户名称	string
app_id	-	当使用APP认证访问时，请求方提供的APP ID	string
time_local	27/Jan/2022:15:56:44	请求时间	string
request_time	0.113	请求延迟	float
request_method	GET	HTTP请求方法	string
scheme	http	请求方式	string
host	c965898968af48248ec7fac4ec0666f4.apic.cn-north-4.huaweicloudapis.com	请求域名	string
router_uri	/api/echo	请求URI	string
server_protocol	HTTP/1.1	请求协议	string
status	200	响应状态码	long
bytes_sent	1443	响应大小（单位：字节，包含状态行、响应头、响应体）	long
request_length	408	请求长度（单位：字节，包含起始行、请求头、请求体）	long
http_user_agent	APIGatewayDebugClient/1.0	用户代理标识	string
http_x_forwarded_for	-	X-Forwarded-For头	string
upstream_addr	100.125.2.39:443	请求的后端地址	string

字段	示例	描述	类型
upstream_uri	/v2/x/fgs/functions/urn:fss:cn-north-4:106506b9a92342df9a5025fc12351cfc:function:default:apigDemo_1640743997661:latest/invocations	请求后端的URI	string
upstream_status	200	后端响应状态码	long
upstream_connect_time	0.01	与后端建立连接所用时间	float
upstream_header_time	0.083	从开始与后端建立连接到从后端获取到首字节所用时间	float
upstream_response_time	0.083	从开始与后端建立连接到从后端获取到最后一个字节所用时间	float
region_id	cn-north-4	云服务区ID	string
all_upstream_response_time	0.083	从开始与后端建立连接到从后端获取到最后一个字节所用时间，单位秒。发生重试时，为所用时间总和	float
errorType	0	API请求的错误类型 ● 0：非流控错误 ● 1：流控错误	long
auth_type	-	API认证类型	string
access_model1	-	认证模式1	string
access_model2	-	认证模式2，开启双重认证时，为自定义认证编号	string
inner_time	0.030000209808 35	apig的内部处理时长，单位秒	float
proxy_protocol_vni	-	VPC终端节点的虚拟网络标识	string
proxy_protocol_vpce_id	-	VPC终端节点的ID	string
proxy_protocol_addr	-	客户端源IP地址	string

字段	示例	描述	类型
body_bytes_sent	486	API请求的Body体大小，单位字节	long
api_name	HttpEchoDemo	API名称	string
app_name	-	当使用APP认证访问时，请求方使用的APP名称	string
provider_app_id	-	API所属的APP ID	string
provider_app_name	-	API所属的APP名称	string
custom_data_log1	-	用户自定义日志字段值1	string
custom_data_log2	-	用户自定义日志字段值2	string
custom_data_log3	-	用户自定义日志字段值3	string
custom_data_log4	-	用户自定义日志字段值4	string
custom_data_log5	-	用户自定义日志字段值5	string
custom_data_log6	-	用户自定义日志字段值6	string
custom_data_log7	-	用户自定义日志字段值7	string
custom_data_log8	-	用户自定义日志字段值8	string
custom_data_log9	-	用户自定义日志字段值9	string
custom_data_log10	-	用户自定义日志字段值10	string
response_source	remote	请求响应来源 ● local：APIG ● remote：后端服务	string
gzip_ratio	0.52	原始响应body体大小与压缩后大小的比率。	float
upstream_scheme	http	后端协议类型。	string
group_id	id	分组ID。	string
apig_err_code	-	网关错误码。	string
function_urn	urn:fss:cn-north-7:6650ff66e4524699bea4c96ff2f268fe:function:default:test123hxy	函数的URN。	string

4.3.4 Astro 轻应用接入 LTS

LTS支持Astro轻应用日志接入，具体接入方法请参见**查看运行日志**。

4.3.5 云堡垒机 CBH 接入 LTS

LTS支持云堡垒机（CBH）日志接入，具体接入方法请参见**配置LTS日志外发服务**。

4.3.6 内容分发网络 CDN 接入 LTS

LTS支持内容分发网络（CDN）日志接入，具体接入方法请参见**实时日志**。

介绍CDN的结构化模板日志详情。

结构化模板日志详情

- 日志示例

表 4-48 结构化模板示例

模板名称	示例日志
CDN	{"request_time":"1666604392000","domain":"findercdn.video.qq.com","method":"GET","scheme":"http","uri":"/BcimRg.txt","uri_param":"cdnkey=******&cdntoken=******&tokenidx=1","client_ip":"192.168.233.142","client_port":"51517","refer_protocol":"-","refer_domain":"-","refer_uri":"-","refer_param":"-","request_size":"301","response_time":"14","response_size":"588","http_code":"403","response_range":"-","request_range":"-","request_body_bytes":"150","content_type":"text/html","hit_info":"HIT","user_agent":"python-requests/2.21.0","uuid":"ce6327e015c1e16f581818b838a6cb0c","via_info":"edge-cache01[14]","xforwordfor":"-"}

说明

cdnkey的值为示例请供参考，请以实际值为准。

- 结构化字段及字段说明

表 4-49 结构化字段

字段	示例	描述	类型
request_time	1666604392000	请求时间戳，单位：毫秒	string
domain	findercdn.video.qq.com	请求的域名	string
method	GET	请求方法	string
scheme	http	请求协议	string
uri	/BcimRg.txt	请求资源	string
uri_param	cdnkey=******&cdntoken=******&tokenidx=1	请求参数	string

字段	示例	描述	类型
client_ip	192.168.233.142	用户真实IP	string
client_port	51517	和CDN节点建连客户端端口	string
refer_protocol	-	HTTP refer中的协议	string
refer_domain	-	HTTP refer中domain信息	string
refer_uri	-	HTTP refer中uri信息	string
refer_param	-	HTTP refer中的参数信息	string
request_size	301	请求大小	string
response_time	14	请求响应时长，单位：毫秒	string
response_size	588	请求返回大小，单位：字节	string
http_code	403	请求响应码	string
response_range	-	应答头里表示的range信息（由源站创建），如bytes：0~99/200	string
request_range	-	用户请求中Header头中range字段取值，如bytes：0~100	string
request_body_bytes	150	实际发送body大小，单位：字节	string
content_type	text/html	请求的资源类型	string
hit_info	HIT	命中信息，取值为HIT（命中）、MISS（未命中）	string
user_agent	python-requests/2.21.0	UA	string
uuid	ce6327e015c1e16f581818b838a6cb0c	请求唯一标识（全网唯一请求ID，即traceid）	string
via_info	edge-cache01[14]	via头信息	string
xforwordfor	-	请求头中XForwardFor字段	string

4.3.7 云防火墙 CFW 接入 LTS

LTS支持云防火墙（CFW）日志接入，具体接入方法请参见**日志配置**。

CFW 流量日志结构化模板日志详情

- CFW流量日志示例日志

表 4-50 结构化模板示例

模板名称	示例日志
CFW流量日志	{"src_port":"60968","to_c_pkts":4,"to_s_bytes":352,"to_c_bytes":1082,"direction":"in2out","bytes":1434,"dst_ip":"100.85.222.231","to_s_pkts":5,"src_ip":"100.95.156.206","dst_host":"www.test1.com","end_time":1695818474000,"protocol":"TCP","dst_port":"80","app":"HTTP","start_time":1695818472000,"vsys":"1","log_type":"internet","packets":9,"dst_region_name":"Chinese Mainland","src_region_name":"Bulgaria","src_region_id":"BG","dst_region_id":"CN","src_vpc":"d644a32b-5c68-4925-bce6-58528da01df0","dst_vpc":"e9ec6dd9-241d-46d5-ae55-47f8c829674f"}

- 结构化字段及字段说明

表 4-51 结构化字段

字段	示例	描述	类型
src_port	60968	源IP地址。	string
to_c_pkts	4	服务端向客户端发送的报文数。	long
to_s_bytes	352	客户端向服务端发送的字节数。	long
to_c_bytes	1082	服务端向客户端发送的字节数。	long
direction	in2out	流量方向。 ● out2in：入方向 ● in2out：出方向	string
bytes	1434	防护流量的字节数。	long
dst_ip	100.85.222.231	目的IP地址。	string
to_s_pkts	5	客户端向服务端发送的报文数。	5ong
src_ip	100.95.156.206	源IP地址	string
dst_host	www.test1.com	目的域名。	string
end_time	1695818474000	流结束时间。	long
protocol	TCP	协议类型	string
dst_port	80	目的端口号	string
app	HTTP	应用类型	string
start_time	1695818472000	流开始时间。	long

字段	示例	描述	类型
vsys	1	防火墙防护方向	string
log_type	internet	日志类型。 • internet：互联网边界流量日志。 • nat：NAT边界流量日志。 • vpc：VPC间流量日志。	string
packets	9	防护流量的报文数。	long
dst_region_name	Chinese Mainland	目的地区名称。	string
src_region_name	Bulgaria	源地区名称。	string
src_region_id	BG	源地区ID。	string
dst_region_id	CN	目的地区ID。	string
src_vpc	d644a32b-5c68-4925-bce6-58528da01df0	源IP地址所在VPC的ID。	string
dst_vpc	e9ec6dd9-241d-46d5-ae55-47f8c829674f	目的IP地址所在VPC的ID。	string

CFW 攻击日志结构化模板日志详情

- CFW攻击日志示例日志

表 4-52 结构化模板示例

模板名称	示例日志
CFW攻击日志	{"attack_rule":"Apache Flink Directory Traversal Vulnerability (CVE-2020-17519)","vsys":"1","source":"predefined","dst_ip":"100.85.222.231","action":"deny","src_port":"51090","attack_rule_id":"331978","direction":"in2out","src_ip":"100.95.156.206","dst_port":"80","log_type":"internet","attack_type":"Vulnerability Exploit Attack","app":"HTTP","event_time":1696652275000,"level":"CRITICAL","packet":"R0VUIC9qb2JtYW5hZ2VyL2xvZ3MvLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmZXRjJTI1MmZwYXNzd2QgSFRUUC8xLjENClVzZXItQWdlbnQ6IGN1cmwvNy4yOS4wDQpIb3N0OiAxMDAuODUuMjIyLjIzMQ0KQWNjZXB0OiAqLyoNCg0K","protocol":"TCP","dst_region_name":"Chinese Mainland","src_region_name":"Chinese Mainland","dst_region_id":"CN","src_region_id":"CN"}

- 结构化字段及字段说明

表 4-53 结构化字段

字段	示例	描述	类型
attack_rule	Apache Flink Directory Traversal Vulnerability (CVE-2020-17519)	检测到攻击的防御规则。	string
vsys	1	防火墙防护方向。	string
source	predefined	检测到攻击的防御模式。 ● 0：基础防御 ● 1：虚拟补丁	string
dst_ip	100.85.222.231	目的IP地址。	string
action	deny	云防火墙当前的响应动作。 ● permit：放行 ● deny：阻断	string
src_port	51090	源端口号。	string
attack_rule_id	331978	检测到攻击的防御规则ID号。	string
direction	in2out	流量方向 ● out2in：入方向 ● in2out：出方向	string
src_ip	100.95.156.20	源IP地址	string
dst_port	80	目的端口号。	string
log_type	internet	日志类型。	string

字段	示例	描述	类型
attack_type	Vulnerability Exploit Attack	发生攻击的类型。 • Vulnerability Exploit Attack：漏洞攻击 • Vulnerability Scan：漏洞扫描 • Trojan：木马病毒 • Worm：蠕虫病毒 • Phishing：网络钓鱼攻击 • Web Attack：Web攻击 • Application DDoS：DDoS攻击 • Buffer Overflow：缓冲区溢出攻击 • Password Attack：密码攻击 • Mail：邮件相关类型的攻击行为 • Access Control：访问控制行为 • Hacking Tool：黑客工具 • Hijacking：劫持行为 • Protocol Exception：存在异常协议 • Spam：存在垃圾邮件 • Spyware：存在间谍软件 • DDoS Flood：DDoS泛洪攻击 • Suspicious DNS Activity：可疑DNS活动 • Other Suspicious Behavior：其他可疑行为	string
app	HTTP	应用类型。	string
event_time	1696652275000	检测到的攻击时间。	long

字段	示例	描述	类型
level	CRITICAL	表示检测到威胁的等级。 ● CRITICAL：严重 ● HIGH：高 ● MIDDLE：中 ● LOW：低	string
packet	R0VUIC9qb2JtYW5hZ2VyL2xvZ3MvLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmLi4lMjUyZi4uJTI1MmYuLiUyNTJmZXRjJTI1MmZwYXNzd2QgSFRUUC8xLjENClVzZXItQWdlbnQ6IGN1cmwvNy4yOS4wDQpIb3N0OiAxMDAuODUuMjIyLjIzMQ0KQWNjZXB0OiAqLyoNCg0K	防护流量的报文数。	string
protocol	TCP	协议类型。	string
dst_region_name	Chinese Mainland	目的地区名称。	string
src_region_name	Chinese Mainland	源地区名称。	string
dst_region_id	CN	目的地区ID。	string
src_region_id	CN	源地区ID。	string

CFW 访问控制日志结构化模板日志详情

- CFW访问控制日志示例

表 4-54 结构化模板示例

模板名称	示例日志
CFW访问控制日志	{"src_port":"47934","protocol":"TCP","dst_port":"80","app":"HTTP","action":"permit","direction":"in2out","rule_id":"d5ad0364-b615-4ca3-ac59-30298cc511d6","vsys":"1","dst_ip":"100.95.152.37","log_type":"internet","hit_time":1695854983000,"src_ip":"100.95.149.249","dst_region_name":"Chinese Mainland","src_region_name":"Chinese Mainland","dst_host":"repo.huaweicloud.com","dst_region_id":"CN","src_region_id":"CN"}

- 结构化字段及字段说明

表 4-55 结构化字段

字段	示例	描述	类型
src_port	47934	源端口号。	string
protocol	TCP	协议类型。	string
dst_port	80	目的端口号。	string
app	HTTP	应用类型。	string
action	permit	防火墙当前的响应动作。 ● permit：放行 ● deny：阻断	string
direction	out2in	流量方向。 ● out2in：入方向 ● in2out：出方向	string
rule_id	d5ad0364-b615-4ca3-ac59-30298cc511d6	触发规则的ID。	string
vsys	1	防火墙防护方向。	string
dst_ip	100.95.152.37	目的IP地址。	string
log_type	internet	日志类型。 ● internet：互联网边界流量日志 ● nat：NAT边界流量日志 ● vpc：VPC间流量日志	string
hit_time	1695854983000	访问发生的时间。	long
src_ip	100.95.149.249	源IP地址。	string
dst_region_name	Chinese Mainland	目的地区名称。	string
src_region_name	Chinese Mainland	源地区名称。	string
dst_host	repo.huaweicloud.com	目的域名。	string
dst_region_id	CN	目的地区ID。	string
src_region_id	CN	源地区ID。	string

4.3.8 云审计服务 CTS 接入 LTS

云日志服务支持云审计服务（Cloud Trace Service）日志接入。

前提条件

购买并使用华为云CTS实例。

操作步骤

云日志服务接入方式为云审计 CTS时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，单击“云审计 CTS”进行CTS接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步”： CTS配置。

步骤4 CTS配置。

单击“前往CTS配置”。

1. 在云审计服务控制台，选择左侧导航树的“追踪器”，进入追踪器信息页面。
2. 单击“创建追踪器”，配置各参数信息。
3. 完成后，单击“确定”。

说明

具体的操作步骤及参数配置，请见**云审计服务《用户指南》**。

步骤5 单击下一步：日志流配置。

表 4-56 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启  该按钮，自动对日志流进行结构化配置和索引配置。日志流结构化配置为CTS系统模板；索引配置为所有CTS解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对CTS日志进行SQL分析，并提供可视化图表。

步骤6 完成

----结束

结构化模板日志详情

- 结构化模板示例

表 4-57 结构化模板示例

模板名称	示例日志
CTS	{"code":"201","source_ip":"10.10.1.10","trace_type":"ApiCall","event_type":"global","project_id":"221123nsada3sda3231das3111ndsab","trace_id":"1eesdd-dad6-11dsaea-edaxfeqdf","trace_name":"demodemodemo","resource_type":"token","trace_rating":"normal","service_type":"IAM","resource_id":"98763hkjhdteoi03861732hjh7983bhd","tracker_name":"global","time":"1597042369296","resource_name":"demodemodemo/demo","record_time":"1597042370464","user":{"domain":{"name":"testdemo","id":"21185d8818e443e1ryjkh71622f09212b"},"name":"testdemo/demo","id":"6hfakl86faqw87dsasasadf09ajbml"}}

- 结构化字段及字段说明

表 4-58 结构化字段

字段	示例	描述	类型
code	201	事件http返回码例如200,400	long
event_type	global	事件类型	string
project_id	221123nsada3sda3231das3111ndsab	项目Id	string
record_time	1597042370464	记录操作的时间，表示方式为时间戳	long
resource_id	98763hkjhdteoi03861732hjh7983bhd	资源的唯一标识	string
resource_name	demodemodemo/demo	资源名称	string
resource_type	token	资源类型	string
service_type	IAM	操作来源	string
source_ip	10.10.1.10	发起本次操作的用户的IP，若为系统内调用，则为空	string

字段	示例	描述	类型
time	1597042369296	事件发生时间。以当地标准时间（采用格林威治时间加当地时区形式）进行展示，例如：2016/12/08 11:24:04 GMT+08:00。在接口中，该字段以时间戳格式进行传输和存储。该字段为格林威治时间1970年01月01日00时00分00秒至现在的总毫秒数	long
trace_id	1eesdd-dad6-11dsaea-edaxfeqdf	操作的唯一标识	string
trace_name	demodemodemo	操作名称	string
trace_rating	normal	操作事件等级，分为normal（正常）、warning（警告）和incident（事故）。 ● normal：代表本次操作成功。 ● warning：代表本次操作失败。 ● incident：代表本次操作引起了比失败更严重的后果，比如会造成节点故障或用户业务故障等情况。	string
trace_type	ApiCall	操作类型，分为如下五种： ● ConsoleAction表示通过管理控制台执行的操作。 ● SystemAction表示系统内部触发的操作。 ● ApiCall表示调用ApiGateway触发的操作。 ● ObsSDK表示通过调用OBS提供的SDK触发的关于OBS桶相关操作。 ● Others表示除去通过“ObsSDK”触发的关于OBS桶相关的操作	string
tracker_name	global	跟踪器名称	string

字段	示例	描述	类型
user.domain.id	21185d8818e443e1ryjkh71622f09212b	用户的domainId	string
user.domain.name	testdemo	用户的domainName	string
user.id	6hfakl86faqw87dsasasadf09ajbml	用户Id	string
user.name	testdemo/demo	用户名称	string

4.3.9 分布式缓存服务 DCS 接入 LTS

LTS支持分布式缓存服务（DCS）日志接入，具体接入方法请参见**审计日志**。

介绍DCS的结构化模板日志详情。

结构化模板日志详情

- DCS审计日志结构化模板示例

表 4-59 结构化模板示例

模板名称	示例日志
DCS审计日志	{"time": 1640966500017, "instance_id": "199a1e5a-8a37-40b9-899e-0ab6805c69eb", "server_addr": "192.168.0.1", "role": "proxy", "client_addr": "10.0.0.1", "client_type": "0", "user": "default", "db": 1, "command_name": "DEL", "command_type": "string", "command_keys": ["key1", "key2", "key3"], "command_param": "DEL key1 key2 key3", "use_time": 500, "extend": ""}

- 结构化字段及字段说明

表 4-60 结构化字段

字段	示例	描述	类型
time	1640966500017	时间	long
instance_id	199a1e5a-8a37-40b9-899e-0ab6805c69eb	实例的ID	string
server_addr	192.168.0.1	服务端IP	string
role	proxy	节点角色	string
client_addr	10.0.0.1	客户端IP	string
client_type	0	客户端类型	string
user	default	账号名	string

字段	示例	描述	类型
db	1	数据库DB	long
command_name	DEL	执行命令	string
command_type	string	命令类型	string
command_keys	[\"key1\",\"key2\",\"key3\"]	命令KEYS	string
command_param	DEL key1 key2 key3	命令内容	string
use_time	500	执行耗时	long
extend	-	扩展信息	string

4.3.10 文档数据库服务 DDS 接入 LTS

LTS支持文档数据库服务（DDS）日志接入，具体接入方法请参见**日志配置管理**。

DDS 慢日志结构化模板日志详情

- DDS慢日志示例日志

表 4-61 结构化模板示例

模板名称	示例日志
DDS慢日志	{"log_type": "slow_log","log_time": "2022-08-20T10:04:03.204000Z","namespace": "data0820.table","database": "data0820","collection": "table","operate_type": "insert","docs_scanned": 0,"docs_returned": 0,"n_deleted": 0,"n_matched": 0,"n_modified": 0,"n_inserted": 10,"cost_time": 555,"lock_time": 0,"whole_message": "{\"op\": \"insert\", \"ns\": \"data0820.usrtable\", \"command\": \"{N}\", \"ninserted\": 1, \"keysInserted\": 1, \"numYield\": 0, \"locks\": {\"Global\": {\"acquireCount\": {\"r\": 5, \"w\": 5}}, \"Database\": {\"acquireCount\": {\"w\": 4, \"W\": 1}}, \"Collection\": {\"acquireCount\": {\"w\": 2}}, \"oplog\": {\"acquireCount\": {\"w\": 2}}}, \"responseLength\": 230, \"protocol\": \"op_msg\", \"millis\": 555, \"ts\": {\"$date\": 1660989843204}, \"client\": \"192.168.0.64\", \"appName\": \"MongoDB Shell\", \"allUsers\": [{\"user\": \"rwuser\", \"db\": \"admin\"}], \"user\": \"rwuser@admin\"}","instance_id": "5b67dc63ba824145aae1f12ff51e58b8in02","node_id": "686a791e690e4db3af591ec4b6f72916no02","client_ip": "190.12.11.11","username": "user中文","keys_examined": 0}

- 结构化字段及字段说明

表 4-62 结构化字段

字段	示例	描述	类型
log_type	slow_log	日志类型	string

字段	示例	描述	类型
log_time	2022-08-20T10:04:03.204000Z	慢日志产生时间	string
namespace	data0820.table	命名空间	string
database	data0820	库名称	string
collection	table	表名称	string
operate_type	insert	操作类型	string
docs_scanned	0	数据库扫描行数	long
docs_returned	0	查询返回结果行数	long
n_deleted	0	删除行数	long
n_matched	0	更新匹配行数	long
n_modified	0	实际更新行数	long
n_inserted	10	插入行数	long
cost_time	555	操作花费的时间	long
lock_time	0	wait_lock的时间	long

字段	示例	描述	类型
whole_message	{\"op\": \"insert\", \"ns\": \"data0820.usrtable\", \"command\": \"{N}\", \"ninserted\": 1, \"keysInserted\": 1, \"numYield\": 0, \"locks\": {\"Global\": {\"acquireCount\": {\"r\": 5, \"w\": 5}}, \"Database\": {\"acquireCount\": {\"w\": 4, \"W\": 1}}, \"Collection\": {\"acquireCount\": {\"w\": 2}}, \"oplog\": {\"acquireCount\": {\"w\": 2}}}, \"responseLength\": 230, \"protocol\": \"op_msg\", \"millis\": 555, \"ts\": {\"$date\": 1660989843204}, \"client\": \"192.168.0.64\", \"appName\": \"MongoDB Shell\", \"allUsers\": [{\"user\": \"rwuser\", \"db\": \"admin\"}], \"user\": \"rwuser@admin\"}	原始日志信息	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例ID	string
node_id	686a791e690e4db3af591ec4b6f72916no02	节点ID	string

DDS 错误日志结构化模板日志详情

- DDS错误日志示例日志

表 4-63 结构化模板示例

模板名称	示例日志
DDS错误日志	{"log_type": "error_log","severity": "Error","log_time": "2022-08-22T09:33:15.142+0000","raw_message": "E QUERY [ClusterDisasterBackupChangeJob] Get global setting disasterBackup failed.","instance_id": "5b67dc63ba824145aae1f12ff51e58b8in02","node_id": "686a791e690e4db3af591ec4b6f72916no02"}

- 结构化字段及字段说明

表 4-64 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	Error	日志级别	string
log_time	2022-08-22T09:33:15.142+0000	日志产生时间	string
raw_message	E QUERY [ClusterDisasterBackupChangeJob] Get global setting disasterBackup failed.	日志内容	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例Id	string
node_id	686a791e690e4db3af591ec4b6f72916no02	节点Id	string

DDS 审计日志结构化模板日志详情

- DDS审计日志示例

表 4-65 结构化模板示例

模板名称	示例日志
DDS审计日志	{"topic":"auditLog","instanceid":"9fbf813bc27e4a3ab54bddf783a4f774in01","nodeid":"bf4cb0413d0b4221be94b08471708586no01","db":"test","coll":"testCollection","optype":"update","time":"2022-08-05T08:24:15.536+0000","user_ip":"10.4.23.205","user_port":"47668","user":"rw_testuser","param":{"command":"update","ns":"test.testCollection","op":[{"q":{"vin":"LDP31B965NG036174"},"u":{"$set":{"timestamp":{"$numberLong":"1659687855535"},"deviceTime":{"$numberLong":"1659687855340"},"longitude":"119.35516805555555","latitude":"26.057936388888891","location":"119.35516805555555,26.057936388888891","height":"10.097286797128618","direction":"12","speed":14,"accuracy":"0","h3Address7":"8741b5300ffffff"}},"upsert":true}],"args":{"update":"testCollection","ordered":true,"$db":"test","$clusterTime":{"clusterTime":{"$timestamp":{"t":1659687855,"i":1685}},"signature":{"hash":{"$binary":"CP5bfEf+gBJZdAxCKtF9HiSeqQY=","$type":"00"},"keyId":{"$numberLong":"7102408879899674942"}}},"lsid":{"id":{"$binary":"PXVVrbuvRuGkypCbu/oXXQ==","$type":"04"}}}}}

- 结构化字段及字段说明

表 4-66 结构化字段

字段	示例	描述	类型
topic	auditLog	消息主题	string
instanceid	9fbf813bc27e4a3ab54bddf783a4f774in01	实例ID	string
nodeid	bf4cb0413d0b4221be94b08471708586no01	节点ID	string
db	test	数据库名	string
coll	testCollection	开启分片的集合名	string
optype	update	操作类型	string
time	2022-08-05T08:24:15.536+0000	审计日志时间	string
user_ip	10.4.23.205	客户端IP	string
user_port	47668	客户端端口	string
user	rw_testuser	数据库用户账户	string

字段	示例	描述	类型
param	{\"command\":\"update\",\"ns\":\"test.testCollection\",\"op\":[{\"q\":{\"vin\":\"LDP31B965NG036174\"},\"u\":{\"$set\":{\"timestamp\":{\"$numberLong\":\"1659687855535\"},\"deviceTime\":{\"$numberLong\":\"1659687855340\"},\"longitude\":\"119.35516805555555\",\"latitude\":\"26.057936388888891\",\"location\":\"119.35516805555555,26.057936388888891\",\"height\":\"10.097286797128618\",\"direction\":\"12\",\"speed\":14,\"accuracy\":\"0\",\"h3Address7\":\"8741b5300ffffff\"}},\"upsert\":true}],\"args\":{\"update\":\"testCollection\",\"ordered\":true,\"$db\":\"test\",\"$clusterTime\":{\"clusterTime\":{\"$timestamp\":{\"t\":1659687855,\"i\":1685}},\"signature\":{\"hash\":{\"$binary\":\"CP5bfEf+gBJZdAxCKtF9HiSeqQY=\",\"$type\":\"00\"},\"keyId\":{\"$numberLong\":\"7102408879899674942\"}}},\"lsid\":{\"id\":{\"$binary\":\"PXVVrbuvRuGkypCbu/oXXQ==\",\"$type\":\"04\"}}}}	数据库操作详情和参数	string

4.3.11 DDoS 防护 AAD 接入 LTS

LTS支持DDoS防护 AAD日志接入，具体接入方法请参见**配置Anti-DDoS日志**。

4.3.12 分布式消息服务 Kafka 版接入 LTS

LTS支持分布式消息服务Kafka日志接入，具体接入方法请参见**查看重平衡日志**。

结构化模板日志详情

- DMS重平衡日志示例日志

表 4-67 结构化模板示例

模板名称	示例日志
DMS重平衡日志	{ "level":"INFO", "timestamp":"2023-03-23 17:23:22,906", "message": { "leaderId":"consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635", "generationId":"1","reason":"Assignment received from leader for group KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0 for generation 1", "groupId":"KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0","coordinatorId":"0", "type":"END_REBALANCE","group":"GroupMetadata(groupId=KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0, generation=1, protocolType=Some(consumer), currentState=CompletingRebalance, members=Map(consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635 -> MemberMetadata(memberId=consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635, clientId=consumer-1, clientHost=/172.31.2.168, sessionTimeoutMs=10000, rebalanceTimeoutMs=300000, supportedProtocols=List(range),)))"} }

- 结构化字段及字段说明

表 4-68 结构化字段

字段	示例	描述	类型
level	INFO	重平衡日志的等级	string
timestamp	2023-03-23 17:23:22,906	Rebalance时间	string
message.leaderId	consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635	消费者Leader ID	string
message.generationId	1	消费者组Generation ID。Generation等同于消费组执行Rebalance的次数，每次Rebalance完成后，Generation都会增加1。	string
message.reason	Assignment received from leader for group KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0 for generation 1	触发Rebalance的原因	string
message.groupId	KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0	消费组ID	string

字段	示例	描述	类型
message.coordinatorId	0	Coordinator组件所在的Broker	string
message.type	END_REBALANCE	触发Rebalance的操作	string
message.group	GroupMetadata(groupId=KMOffsetCache-dms-vm-fa3cf9d6-manager-shared-server-0, generation=1, protocolType=Some(consumer), currentState=CompletingRebalance, members=Map(consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635 -> MemberMetadata(memberId=consumer-1-177817b6-1f29-4717-8a83-dda8eaab1635, clientId=consumer-1, clientHost=/172.31.2.168, sessionTimeoutMs=10000, rebalanceTimeoutMs=300000, supportedProtocols=List(range),)))	消费组中消费者的信息	string

4.3.13 数据复制服务 DRS 接入 LTS

LTS支持数据复制服务（ DRS ）日志接入，具体接入方法请参见**日志配置管理**。

4.3.14 数据仓库服务 GaussDB(DWS)接入 LTS

LTS支持数据仓库GaussDB（ DWS ）日志接入，具体接入方法请参见**集群日志管理**。

4.3.15 弹性负载均衡 ELB 接入 LTS

云日志服务支持弹性负载均衡（ Elastic Load Balance ）日志接入。

前提条件

创建并使用华为云ELB实例。

使用限制

当前ELB日志仅支持七层独享型负载均衡和七层共享型负载均衡，不支持四层共享型负载均衡。

设置 ELB 接入

云日志服务接入方式为负载均衡 ELB时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，单击“负载均衡 ELB”进行ELB接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步”： ELB配置。

步骤4 ELB配置。

单击“前往ELB配置”。

1. 在网络控制台，选择弹性负载均衡下的“负载均衡器”。
2. 在弹性负载均衡页面中，单击待操作的负载均衡器“名称/ID”，进入详情页面。
3. 选择“访问日志”，单击“配置访问日志”。
4. 在弹出的配置访问日志页面中，选择对应的日志组和日志流。

说明

具体的操作步骤，请见**弹性负载均衡《用户指南》**。

步骤5 日志流配置。

表 4-69 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启  该按钮，自动对日志流进行结构化配置和索引配置。日志流结构化配置为ELB系统模板；索引配置为所有ELB解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对ELB日志进行SQL分析，并提供可视化图表。
自动为日志流添加标签：log_type=elb_7layer_access	开启该按钮，自动为日志流添加标签（log_type=elb_7layer_access）后，ELB仪表盘模板才能匹配该日志流。
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建ELB仪表盘。

步骤6 完成。

----结束

结构化模板日志详情

- ELB示例日志

表 4-70 结构化模板示例

模板名称	示例日志
ELB	1594727856.337 e7c37d97-e922-457c-bbf3-dsadeqac [2020-07-14T19:57:36+08:00] elb_01 192.0.0.0:88888 200 "GET http://prod.sss.ads.sg2.aaa/loc/ation?version=3&ip=100.0.0.0&coordinate=27.7044784,85.3007481&device_id=dsadsadasdsadasd&beyla_id=wqeb123ndadsa233ddada HTTP/1.1" 233 293 138 0.001 "200" "0.000" "0.001" "0.001" "100.0.0.0:9999" "lua-resty-http/0.14 (Lua) ngx_lua/10000" "-" "-" loadbalancer_edsaee-4c9c-b467-5b8126b2f7f7dsa listener_6077809b-913f-466d-a96c-376f08882d5d 08cc2b3f68aa4dddd1e6a90dddd1688348a4480 pool_b2f2966c-043d-4674-ad4b-c15f2adb2c6b "-" 2fb78dsadadq1213das1121dab146ad3cb0 -:80 "101.0.0.0:10000" - - - - 9739

- 结构化字段及字段说明

表 4-71 结构化字段

字段	示例	描述	类型
msec	1594727856.337	以秒为单位的时间，日志写入时的分辨率为毫秒。	float
access_log_topic_id	e7c37d97-e922-457c-bbf3-dsadeqac	访问日志流ID	string
time_iso8601	2020-07-14T19:57:36+08:00	日志写入时的时间，采用ISO 8601标准格式本地时间	string
log_ver	elb_01	ELB服务日志版本号	string
remote_addr	192.0.0.0	客户端IP地址	string
remote_port	88888	客户端端口	long
status	200	ELB响应的状态码。	long
request_method	GET	请求方法	string
scheme	http	请求方式	string
host	prod.sss.ads.sg2.aaa	主机名	string
router_request_uri	location?version=3&ip=100.0.0.0&coordinate=27.7044784,85.3007481&device_id=dsadsadasdsadasd&beyla_id=wqeb123ndadsa233ddada	请求uri	string

字段	示例	描述	类型
server_protocol	HTTP/1.1	请求协议	string
request_length	233	从客户端收到的请求长度（包括请求header和请求body）	long
bytes_sent	293	发送到客户端的字节数	long
body_bytes_sent	138	发送到客户端的字节数（不包括响应头）	long
request_time	0.001	请求处理时间，即ELB收到第一个客户端请求报文到ELB发送完响应报文的时间间隔（单位：秒	float
upstream_status	200	从上游服务器获得的响应状态码，当ELB代理进行请求重试时会包含多个响应的状态码，当请求未被正确转发到后端服务器时此字段为 -	long
upstream_connect_time	0.000	与上游服务器建立连接所花费的时间，时间以秒为单位，分辨率为毫秒。当ELB代理进行请求重试时会包含多个连接的时间，当请求未被正确转发到后端服务器时此字段为 -	float
upstream_header_time	0.001	从上游服务器接收响应头所花费的时间，时间以秒为单位，分辨率为毫秒。当ELB代理进行请求重试时会包含多个响应时间，当请求未被正确转发到后端服务器时此字段为 -	float
upstream_response_time	0.001	从上游服务器接收响应所花费的时间，时间以秒为单位，分辨率为毫秒。当ELB代理进行请求重试时会包含多个响应时间，当请求未被正确转发到后端服务器时此字段为 -	float

字段	示例	描述	类型
upstream_addr	100.0.0.0:9999	后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。 （该参数适用于独享型负载均衡）	string
http_user_agent	lua-resty-http/0.14 (Lua) ngx_lua/10000	ELB收到请求头中的http_user_agent内容，表示客户端的系统型号、浏览器信息等。	string
http_referer	-	ELB收到请求头中的http_referer内容，表示该请求所在的页面链接。	string
http_x_forwarded_for	-	ELB收到请求头中的http_x_forwarded_for内容，表示请求经过的代理服务器IP地址。	string
lb_name	loadbalancer_edsaee-4c9c-b467-5b8126b2f7f7dsa	负载均衡器的名称（格式为“loadbalancer_”+“负载均衡器ID”）	string
listener_name	listener_6077809b-913f-466d-a96c-376f08882d5d	监听器的名称（格式为“listener_”+“监听器ID”）	string
listener_id	08cc2b3f68aa4dddd1e6a90dddd1688348a4480	监听器在ELB服务内部的ID（客户可忽略）	string
pool_name	pool_b2f2966c-043d-4674-ad4b-c15f2adb2c6b	后端服务器组名称（格式为“pool_”+“后端服务器组ID”）	string
member_name	-	后端服务器的名称（格式为“member_”+“服务器ID”，尚未支持）。可能有多个值，每个值都是member_id或者-，用逗号空格隔开。	string
tenant_id	2fb78dsadadq1213das1121dab146ad3cb0	租户ID	string
eip_address	-	弹性IP地址	string
eip_port	80	监听器监听的端口号	long

字段	示例	描述	类型
upstream_addr_priv	101.0.0.0:10000	后端主机的IP地址和端口号。可能有多个值，每个值都是ip:port或者-，用逗号空格隔开。 （该参数适用于共享型负载均衡）	string
certificate_id	-	[HTTPS监听器]SSL连接建立时使用的证书ID	string
ssl_protocol	-	[HTTPS监听器]SSL连接建立使用的协议，非HTTPS监听器，此字段为 -	string
ssl_cipher	-	[HTTPS监听器]SSL连接建立使用的加密套件，非HTTPS监听器，此字段为 -	string
sni_domain_name	-	[HTTPS监听器]SSL握手时客户端提供的SNI域名，非HTTPS监听器，此字段为 -	string
tcpinfo_rtt	9739	ELB与客户端之间的tcp rtt时间，单位：微秒	long

4.3.16 企业路由器 ER 接入 LTS

LTS支持ER企业路由器日志接入，具体接入方法请参见创建流日志。

结构化模板日志详情

- ER企业路由器日志示例

  表 4-72 结构化模板示例

模板名称	示例日志
ER企业路由器	1 0ebc7f641d80f4d32f9dc0056bdaae5b 740cc7e2-c686-496e-9fae-2dfc33e9d443 2756dc58-49be-413d-852f-accc5657a6c3 192.168.3.227 172.16.0.206 0 0 1 279 27342 1670338595 1670339195 egress

- 结构化字段及字段说明

  表 4-73 结构化字段

字段	示例	描述	类型
version	1	ER流日志版本	long

字段	示例	描述	类型
project_id	0ebc7f641d80f4d32f9dc0056bdaae5b	项目ID	string
instance_id	740cc7e2-c686-496e-9fae-2dfc33e9d443	ER实例的ID	string
resource_id	2756dc58-49be-413d-852f-accc5657a6c3	流量所属连接的ID	string
srcaddr	192.168.3.227	源地址	string
dstaddr	172.16.0.206	目的地址	string
srcport	0	源端口	long
dstport	0	目的端口	long
protocol	1	IANA协议编号 关于协议的更多信息，请参见**Internet协议编号**	long
packets	279	本段流日志捕获窗口时间周期内数据包的数量	long
bytes	27342	本段流日志捕获窗口时间周期内数据包的大小	long
start	1670338595	本段流日志捕获窗口启动的时间，采用Unix秒的格式	long
end	1670339195	本段流日志捕获窗口结束的时间，采用Unix秒的格式	long
direct	egress	流量的方向： • ingress：入方向，表示流量从外部进入ER连接内 • egress：出方向，表示流量从ER连接发送出去	string

4.3.17 函数工作流 FunctionGraph 接入 LTS

LTS支持函数工作流（ FunctionGraph ）日志接入，具体接入方法请参见**管理函数日志**。

4.3.18 云数据库 GaussDB 接入 LTS

LTS支持云数据库 GaussDB日志接入，具体接入方法请参见**LTS日志**。

结构化模板日志详情

- GAUSSV5审计日志示例

表 4-74 结构化模板示例

模板名称	示例日志
GAUSSV5审计日志	{ "username": "rdsAdmin", "client_conninfo": "cm_agent@10.254.95.70", "instanceId": "96e86f462bbc4f2286d7c8274815d0fein14", "detail_info": "xid=30818, SET statement_timeout = 10000000;n", "thread_id": "140463114942208@713872403507507", "result": "ok", "database": "postgres", "local_port": "8001", "userid": "10", "nodeId": "06c267fad8054a0abcb17cfa3b8f260cno14", "node_name": "dn_6001_6002_6003", "object_name": "statement_timeout", "time": "2022-08-15 17:53:23+08", type: "set_parameter", "remote_port": "50952" }

- 结构化字段及字段说明

表 4-75 结构化字段

字段	示例	描述	类型
username	rdsAdmin	用户名	string
client_conninfo	cm_agent@10.254.95.70	客户端连接信息	string
instanceId	96e86f462bbc4f2286d7c8274815d0fein14	实例ID	string
detail_info	xid=30818, SET statement_timeout = 10000000;n	执行的SQL语句	string
thread_id	140463114942208@713872403507507	线程ID	string
result	ok	SQL执行结果	string
database	postgres	使用的数据库	string
local_port	8001	本地端口	string
userid	10	用户ID	string
nodeId	06c267fad8054a0abcb17cfa3b8f260cno14	节点ID	string
node_name	dn_6001_6002_6003	节点名称	string
object_name	statement_timeout	对象的名称	string
time	2022-08-15 17:53:23+08	审计日志时间	string
type	set_parameter	SQL开始执行时间	string

字段	示例	描述	类型
remote_port	50952	远程端口	string

4.3.19 图引擎服务 GES 接入 LTS

LTS支持图引擎服务（GES）日志接入，具体接入方法请参见**自定义创建图**。

4.3.20 云数据库 GaussDB(for MySQL)接入 LTS

LTS支持云数据库 GaussDB(for MySQL)日志接入，具体接入方法请参见**日志配置管理**。

GAUSSDB_MYSQL 慢日志结构化模板日志详情

- GAUSSDB_MYSQL慢日志示例

表 4-76 结构化模板示例

模板名称	示例日志
GAUSSDB_MYSQL慢日志	{"start_time":"2022-07-27T02:49:19.000","user":"commerce","host":"100.*.*.222","query_time":"1.461583","lock_time":"0.000050","rows_sent":"500","rows_examined":"581000","command_text":"SELECT DN_N.record_id \"a.id\",DN_N.name \"a.name\",DN_N.valueN \"a.ExternalCode\",DN_N.valueN a.DeviceName\",DN_N.valueN \"a.DeviceDef\",DN_N.created_date \"a.createdDate \",DN_N.last_modified_date \"a.lastModifiedDate\",DN_N.valueN \"a.DeviceProduct\",DN_N.valueN \"a.Channel\",DN_N.valueN \"a.Status\",CN_N.valueN \"a.Remark\",DN_N.valueN \"a.NodeId \",DN_N.valueN \"a.ConnectStatus\",CAST(DN_N.valueN AS CHAR(N)) \"a.GatewayId\",CAST(DN_N.valueN AS CHAR(N)) \"a.HMI \",DN_N.valueN \"a.SerialNo\",CAST(DN_N.valueN AS DECIMAL(N,N)) \"a.TelemetryPeriod\",DN_N.valueN \"a.ConnectStatusChgTime \",DN_N.valueN \"a.DeviceNumber\",CAST(DN_N.valueN AS CHAR(N)) \"a.ControllerType\",CAST(DN_N.valueN AS CHAR(N)) \"a.ProjectId \",DN_N.valueN \"a.RegisterStatus\",DN_N.created_date ORD_FN FROM dataN DN_N,clobs CN_N WHERE (DN_N.tenant_id= N AND DN_N.obj_id= N AND DN_N.tenant_id= CN_N.tenant_id AND DN_N.obj_id= CN_N.obj_id AND DN_N.record_id= CN_N.record_id) AND ((DN_N.valueN = N)) ORDER BY DN_N.created_date DESC limit N,N;","database":"saas_perf","log_type":"slow_log","log_time":"1658890159","operate_type":"SELECT"}

- 结构化字段及字段说明

表 4-77 结构化字段

字段	示例	描述	类型
start_time	2022-07-27T02:49:19.000	sql开始执行时间	string
user	commerce	用户名	string
host	100.*.*.222	连接IP	string

字段	示例	描述	类型
query_time	1.461583	sql执行时间	string
lock_time	0.000050	等待锁时间	string
rows_sent	500	查询返回行数	string
rows_examined	581000	查询扫描行数	string

字段	示例	描述	类型
command_text	SELECT DN_N.record_id \"a.id \",DN_N.name \"a.name \",DN_N.valueN \"a.ExternalCode \",DN_N.valueN \"a.DeviceName \",DN_N.valueN \"a.DeviceDef \",DN_N.created_date \"a.createdDate \",DN_N.last_modified_date \"a.lastModifiedDate \",DN_N.valueN \"a.DeviceProduct \",DN_N.valueN \"a.Channel \",DN_N.valueN \"a.Status \",CN_N.valueN \"a.Remark \",DN_N.valueN \"a.NodeId \",DN_N.valueN \"a.ConnectStatus \",CAST(DN_N.valueN AS CHAR(N)) \"a.GatewayId \",CAST(DN_N.valueN AS CHAR(N)) \"a.HMI \",DN_N.valueN \"a.SerialNo \",CAST(DN_N.valueN AS DECIMAL(N,N)) \"a.TelemetryPeriod \",DN_N.valueN \"a.ConnectStatusChgTime\",DN_N.valueN \"a.DeviceNumber \",CAST(DN_N.valueN AS CHAR(N)) \"a.ControllerType \",CAST(DN_N.valueN AS CHAR(N)) \"a.ProjectId \",DN_N.valueN \"a.RegisterStatus \",DN_N.created_date	执行的SQL语句	string

字段	示例	描述	类型
	ORD_FN FROM dataN DN_N,clobs CN_N WHERE (DN_N.tenant_id= N AND DN_N.obj_id= N AND DN_N.tenant_id= CN_N.tenant_id AND DN_N.obj_id= CN_N.obj_id AND DN_N.record_id= CN_N.record_id) AND ((DN_N.valueN = N)) ORDER BY DN_N.created_date DESC limit N,N;		
database	saas_perf	使用的数据库	string
log_type	slow_log	日志类型	string
log_time	1658890159	日志执行时间戳	string
operate_type	SELECT	sql操作类型，例如select，update，insert等	string

GAUSSDB_MYSQL 错误日志结构化模板日志详情

- GAUSSDB_MYSQL错误日志示例

  表 4-78 结构化模板示例

模板名称	示例日志
GAUSSDB_MYSQL错误日志	{"log_type":"error_log","severity":"WARNING","log_time":"2022-08-22T06:52:08Z","raw_message":"Occur error when reading bytes from a network handler. Client actively closes the connection."}

- 结构化字段及字段说明

  表 4-79 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	WARNING	日志级别	string
log_time	2022-08-22T06:52:08Z	错误日志产生时间	string

字段	示例	描述	类型
raw_message	Occur error when reading bytes from a network handler. Client actively closes the connection.	日志内容	string

4.3.21 云数据库 GeminiDB 接入 LTS

LTS支持GeminiDB Redis日志接入，具体接入方法请参见**日志配置管理**。

结构化模板日志详情

- GeminiDB Redis慢日志示例日志

表 4-80 结构化模板示例

模板名称	示例日志
GeminiDB Redis慢日志	{ "instance_id": "32eaaf6c5a0142e3a6d80740cd5b3803in12", "node_id": "597a15b9f2ef4436811c5edcc67c013cno12", "database": "0", "log_type": "slow_log", "operate_type": "sismember", "log_time": "2022-10-12T07:42:21.253484Z", "cost_time": 1277.47, "whole_message": "0g1Oxct"}

- 结构化字段及字段说明

表 4-81 结构化字段

字段	示例	描述	类型
instance_id	32eaaf6c5a0142e3a6d80740cd5b3803in12	实例Id	string
node_id	597a15b9f2ef4436811c5edcc67c013cno12	节点Id	string
database	0	库名称	string
log_type	slow_log	日志类型	string
operate_type	sismember	操作类型	string
log_time	2022-10-12T07:42:21.253484Z	慢日志产生时间	string
cost_time	1277.47	操作花费的时间	float
whole_message	0g1Oxct	原始日志信息	string

4.3.22 云数据库 GeminiDB Mongo 接入 LTS

云数据库GeminiDB Mongo日志对接LTS正在白名单用户内测中，如果您有需要请提工单给GeminiDB云服务开通。详细操作请参考**提交工单**。

GeminiDB Mongo 错误日志结构化模板日志详情

- GeminiDB Mongo错误日志示例

表 4-82 结构化模板示例

模板名称	示例日志
GeminiDB Mongo错误日志	{"log_type":"error_log","severity":"Error","log_time":"2022-08-22T09:33:15.142+0000","raw_message":"E QUERY [ClusterDisasterBackupChangeJob] Get global setting disasterBackup failed.","instance_id":"5b67dc63ba824145aae1f12ff51e58b8in02","node_id":"686a791e690e4db3af591ec4b6f72916no02"}

- 结构化字段及字段说明

表 4-83 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	Error	日志级别	string
log_time	2022-08-22T09:33:15.142+0000	日志产生时间	string
raw_message	E QUERY [ClusterDisasterBackupChangeJob] Get global setting disasterBackup failed.	日志内容	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例Id	string
node_id	686a791e690e4db3af591ec4b6f72916no02	节点Id	string

GeminiDB Mongo 慢日志结构化模板日志详情

- GeminiDB Mongo慢日志示例

表 4-84 结构化模板示例

模板名称	示例日志
GeminiDB Mongo慢日志	{"log_type":"slow_log","log_time":"2022-08-20T10:04:03.204000Z","namespace":"data0820.table","database":"data0820","collection":"table","operate_type":"insert","docs_scanned":0,"docs_returned":0,"n_deleted":0,"n_matched":0,"n_modified":0,"n_inserted":10,"cost_time":555,"lock_time":0,"whole_message":{"op":"insert","ns":"data0820.usrtable","command":"{N}","ninserted":1,"keysInserted":1,"numYield":0,"locks":{"Global":{"acquireCount":{"r":5,"w":5}},"Database":{"acquireCount":{"w":4,"W":1}},"Collection":{"acquireCount":{"w":2}},"oplog":{"acquireCount":{"w":2}}},"responseLength":230,"protocol":"op_msg","millis":555,"ts":{"$date":1660989843204},"client":"192.168.0.64","appName":"MongoDBShell","allUsers":[{"user":"rwuser","db":"admin"}],"user":"rwuser@admin"},"instance_id":"5b67dc63ba824145aae1f12ff51e58b8in02","node_id":"686a791e690e4db3af591ec4b6f72916no02"}

- 结构化字段及字段说明

表 4-85 结构化字段

字段	示例	描述	类型
log_type	slow_log	日志类型	string
log_time	2022-08-20T10:04:03.204000Z	慢日志产生的时间	string
namespace	data0820.table	命名空间	string
database	data0820	库名称	string
collection	table	表名称	string
operate_type	insert	操作类型	string
docs_scanned	0	数据库扫描行数	long
docs_returned	0	查询返回结果行数	long
n_deleted	0	删除行数	long
n_matched	0	更新匹配行数	long
n_modified	0	实际更新行数	long
n_inserted	10	插入行数	long
cost_time	555	操作花费的时间	long
lock_time	0	wait_lock的时间	long

字段	示例	描述	类型
whole_message	{"op":"insert","ns":"data0820.usrtable","command":"{N}","ninserted":1,"keysInserted":1,"numYield":0,"locks":{"Global":{"acquireCount":{"r":5,"w":5}},"Database":{"acquireCount":{"w":4,"W":1}},"Collection":{"acquireCount":{"w":2}},"oplog":{"acquireCount":{"w":2}}},"responseLength":230,"protocol":"op_msg","millis":555,"ts":{"$date":1660989843204},"client":"192.168.0.64","appName":"MongoDBShell","allUsers":[{"user":"rwuser","db":"admin"}],"user":"rwuser@admin"}	原始日志信息	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例ID	string
node_id	686a791e690e4db3af591ec4b6f72916no02	节点ID	string

4.3.23 云数据库 GeminiDB Cassandra 接入 LTS

云数据库GeminiDB Cassandra日志对接LTS正在白名单用户内测中，如果您有需要请提工单给GeminiDB云服务开通。详细操作请参考**提交工单**。

GeminiDB Cassandra 慢日志结构化模板日志详情

- GeminiDB Cassandra慢日志示例

表 4-86 结构化模板示例

模板名称	示例日志
GeminiDB Cassandra慢日志	{"instance_id":"32eaaf6c5a0142e3a6d80740cd5b3803in12","node_id":"597a15b9f2ef4436811c5edcc67c013cno12","keyspace":"test","table":"test","log_type":"slow_log","operate_type":"select","log_time":"2022-10-12T07:42:21.253484Z","cost_time":1277.47,"whole_message":"0g1Oxct"}

- 结构化字段及字段说明

表 4-87 结构化字段

字段	示例	描述	类型
instance_id	32eaaf6c5a0142e3a6d80740cd5b3803in12	实例Id	string
node_id	597a15b9f2ef4436811c5edcc67c013cno12	节点Id	string
keyspace	test	键空间	string
table	test	数据库表名	string
log_type	slow_log	日志类型	string
operate_type	select	操作类型	string
log_time	2022-10-12T07:42:21.253484Z	慢日志产生的时间	string
cost_time	1277.47	操作花费的时间	float
whole_message	0g1Oxct	原始日志信息	string

4.3.24 华为 HiLens 接入 LTS

LTS支持华为HiLens日志接入，具体接入方法请参见**管理设备日志**。

4.3.25 设备接入 IoTDA 接入 LTS

LTS支持设备接入（IoTDA）日志接入，具体接入方法请参见**查看运行日志**。

4.3.26 AI 开发平台 ModelArts 接入 LTS

LTS支持AI开发平台（ModelArts）日志接入，具体接入方法请参见**部署为在线服务**。

4.3.27 MapReduce 服务 MRS 接入 LTS

LTS支持MapReduce服务（MRS）日志接入，具体接入方法请参见**MRS服务对接云日志服务**。

4.3.28 云数据库 RDS for MySQL 接入 LTS

LTS支持云数据库（RDS） for MySQL日志接入，具体接入方法请参见**日志配置管理**。

MYSQL 慢日志结构化模板日志详情

- MYSQL慢日志示例

表 4-88 结构化模板示例

模板名称	示例日志
MYSQL慢日志	{"start_time": "2022-07-27T02:49:19.000","user": "commerce","host": "100.*.*.222","query_time": "1.461583","lock_time": "0.000050","rows_sent": "500","rows_examined": "581000","command_text": "SELECT DN_N.record_id \"a.id \",DN_N.name \"a.name\",DN_N.valueN \"a.ExternalCode \",DN_N.valueN \"a.DeviceName\",DN_N.valueN \"a.DeviceDef \",DN_N.created_date \"a.createdDate\",DN_N.last_modified_date \"a.lastModifiedDate\",DN_N.valueN \"a.DeviceProduct\",DN_N.valueN \"a.Channel\",DN_N.valueN \"a.Status\",CN_N.valueN \"a.Remark \",DN_N.valueN \"a.NodeId\",DN_N.valueN \"a.ConnectStatus \",CAST(DN_N.valueN AS CHAR(N)) \"a.GatewayId \",CAST(DN_N.valueN AS CHAR(N)) \"a.HMI\",DN_N.valueN \"a.SerialNo\",CAST(DN_N.valueN AS DECIMAL(N,N)) \"a.TelemetryPeriod\",DN_N.valueN \"a.ConnectStatusChgTime \",DN_N.valueN \"a.DeviceNumber\",CAST(DN_N.valueN AS CHAR(N)) \"a.ControllerType\",CAST(DN_N.valueN AS CHAR(N)) \"a.ProjectId \",DN_N.valueN \"a.RegisterStatus\",DN_N.created_date ORD_FN FROM dataN DN_N,clobs CN_N WHERE (DN_N.tenant_id= N AND DN_N.obj_id= N AND DN_N.tenant_id= CN_N.tenant_id AND DN_N.obj_id= CN_N.obj_id AND DN_N.record_id= CN_N.record_id) AND ((DN_N.valueN = N)) ORDER BY DN_N.created_date DESC limit N,N;","database": "saas_perf","log_type": "slow_log","log_time": "1658890159","operate_type": "SELECT","node_id":"5d6c61bbd49b4ad3a1572461811e3dacno01","instance_id":"207032924c644f429b74f6fc5d8c97f9in01"}

- 结构化字段及字段说明

表 4-89 结构化字段

字段	示例	描述	类型
start_time	2022-07-27T02:49:19.000	语句开始执行时间	string
user	commerce	用户	string
host	100.*.*.222	主机	string
query_time	1.461583	语句执行时间，以秒为单位	string
lock_time	0.000050	获取锁的时间，以秒为单位	string
rows_sent	500	发送到客户端的行数	string
rows_examined	581000	服务器层检查的行数	string

字段	示例	描述	类型
command_text	SELECT DN_N.record_id \"a.id \",DN_N.name \"a.name \",DN_N.valueN \"a.ExternalCode \",DN_N.valueN \"a.DeviceName \",DN_N.valueN \"a.DeviceDef \",DN_N.created_date \"a.createdDate \",DN_N.last_modified_date \"a.lastModifiedDate \",DN_N.valueN \"a.DeviceProduct \",DN_N.valueN \"a.Channel \",DN_N.valueN \"a.Status \",CN_N.valueN \"a.Remark \",DN_N.valueN \"a.NodeId \",DN_N.valueN \"a.ConnectStatus \",CAST(DN_N.valueN AS CHAR(N)) \"a.GatewayId \",CAST(DN_N.valueN AS CHAR(N)) \"a.HMI \",DN_N.valueN \"a.SerialNo \",CAST(DN_N.valueN AS DECIMAL(N,N)) \"a.TelemetryPeriod \",DN_N.valueN \"a.ConnectStatusChgTime \",DN_N.valueN \"a.DeviceNumber \",CAST(DN_N.valueN AS CHAR(N)) \"a.ControllerType \",CAST(DN_N.valueN AS CHAR(N)) \"a.ProjectId \",DN_N.valueN \"a.RegisterStatus \",DN_N.created_date ORD_FN FROM dataN DN_N,clobs CN_N WHERE (DN_N.tenant_id= N AND DN_N.obj_id= N AND DN_N.tenant_id= CN_N.tenant_id AND DN_N.obj_id= CN_N.obj_id AND DN_N.record_id= CN_N.record_id) AND ((DN_N.valueN = N)) ORDER	sql语句	string

字段	示例	描述	类型
	BY DN_N.created_date DESC limit N,N;		
database	saas_perf	数据库	string
log_type	slow_log	日志类型	string
log_time	1658890159	语句执行结束时间	string
operate_type	SELECT	语句类型	string
node_id	5d6c61bbd49b4ad3a1572461811e3dacno01	节点ID	string
instance_id	207032924c644f429b74f6fc5d8c97f9in01	实例ID	string

MYSQL 错误日志结构化模板日志详情

- MYSQL错误日志示例

表 4-90 结构化模板示例

模板名称	示例日志
MYSQL错误日志	{"log_type": "error_log","severity": "WARNING","log_time": "2022-08-22T06:52:08Z","raw_message": "Occur error when reading bytes from a network handler. Client actively closes the connection.","node_id":"5d6c61bbd49b4ad3a1572461811e3dacno01","instance_id":"207032924c644f429b74f6fc5d8c97f9in01"}

- 结构化字段及字段说明

表 4-91 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	WARNING	事件优先级, 包含system, error, warning, note/information级别事件	string
log_time	2022-08-22T06:52:08Z	发生时间	string
raw_message	Occur error when reading bytes from a network handler. Client actively closes the connection.	事件消息	string

字段	示例	描述	类型
node_id	5d6c61bbd49b4ad3a1572461811e3dacno01	节点ID	string
instance_id	207032924c644f429b74f6fc5d8c97f9in01	实例ID	string

MYSQL 审计日志结构化模板日志详情

- MYSQL审计日志示例

表 4-92 结构化模板示例

模板名称	示例日志
MYSQL审计日志	{"logType":"audit_log","instanceId":"e2a8db82a9d74982a6021c6758d57e00in01","nodeId":"633a1f5a3db9445586f297f9c026b91bno01","record_id":"2","connection_id":"112","connection_status":"0","name":"Query","timestamp":"2023-01-06T06:35:46 UTC","command_class":"show_tables","sqltext":"show tables","user":"root[root] @ [10.58.239.247]","host":"","external_user":"","ip":"10.58.239.247","default_db":""}

- 结构化字段及字段说明

表 4-93 结构化字段

字段	示例	描述	类型
logType	audit_log	日志类型	string
instanceId	e2a8db82a9d74982a6021c6758d57e00in01	实例Id	string
nodeId	633a1f5a3db9445586f297f9c026b91bno01	节点Id	string
record_id	2	审计日志单条记录的记录ID，记录审计日志的每条SQL的唯一global id	string
connection_id	112	该条记录执行的会话ID，与show processlist中的ID一致	string
connection_status	0	会话状态，常见为执行语句的错误返回码，普通执行成功返回0	string

字段	示例	描述	类型
name	Query	记录类型名称，通常情况下dml，ddl操作均为QUERY， 连接断开为CONNECT和QUIT	string
timestamp	2023-01-06T06:35:46 UTC	记录的UTC时间	string
command_class	show_tables	执行的SQL命令类型，内部为解析得到的SQL类型，例如select，update（连接断开不存在该项）	string
sqltext	show tables	执行的SQL具体内容（连接断开审计不存在该项）	string
user	root[root] @ [10.58.239.247]	登录的账户	string
host	-	登录的host，当本地登录时为localhost，远程登录为空	string
external_user	-	代理用户名称	string
ip	10.58.239.247	通过远程连接的客户端IP，本地连接为空	string
default_db	-	执行SQL时默认的数据库	string

4.3.29 云数据库 RDS for PostgreSQL 接入 LTS

LTS支持云数据库（RDS）for PostgreSQL日志接入，具体接入方法请参见**日志配置管理**。

POSTGRESQL 慢日志结构化模板日志详情

- POSTGRESQL慢日志示例

表 4-94 结构化模板示例

模板名称	示例日志
POSTGRESQL慢日志	{"log_type":"slow_log","execute_time":328.662,"user":"authoring","log_time":"2022-07-24T10:06:41.000","database":"authoring-test","statement":"SELECT * FROM (SELECT n.user_id,n.id AS resource_id,e.create_at AS begin_time,e.create_at AS end_time ,N AS resource_spec_code,COALESCE(cast(e.flavor as varchar), cast(s.volume_size as varchar)) AS billing_unit,c.az_id,-N AS accumulate_factor_value,CONCAT(N, s.id, N) AS bss_params,n.project_id, n.domain_id, e.status , N AS resource_type , w.workspace_id,w.enterprise_project_id FROM t_resource_status_event e INNER JOIN t_notebook_evs_storage s on s.id=e.resource_id LEFT JOIN t_notebook_instance n on s.id=n.storage_id LEFT JOIN t_logic_cluster l on n.resource_cluster_id=l.id LEFT JOIN t_cce_cluster c on c.id=l.cce_id LEFT JOIN t_workspace w on w.workspace_id=n.workspace_id WHERE e.category = N AND s.resource_ownership=N AND e.create_at BETWEEN $N AND $N UNION ALL SELECT n.user_id,n.id AS resource_id,$N AS begin_time,$N AS end_time ,N AS resource_spec_code,COALESCE(cast(e.flavor as varchar), cast(s.volume_size as varchar)) AS billing_unit,c.az_id,-N AS accumulate_factor_value,CONCAT(N, s.id, N) AS bss_params,n.project_id, n.domain_id, e.status , N AS resource_type , w.workspace_id,w.enterprise_project_id FROM t_resource_status_event e INNER JOIN t_notebook_evs_storage s on s.id=e.resource_id LEFT JOIN t_notebook_instance n on s.id=n.storage_id LEFT JOIN t_logic_cluster l on n.resource_cluster_id=l.id LEFT JOIN t_cce_cluster c on c.id=l.cce_id LEFT JOIN t_workspace w on w.workspace_id=n.workspace_id INNER JOIN (SELECT resource_id,max(create_at) as create_at FROM t_resource_status_event WHERE create_at < $N AND category = N GROUP BY resource_id) x ON e.resource_id=x.resource_id AND e.create_at=x.create_at WHERE e.create_at < $N AND e.category = N AND e.status = N AND s.resource_ownership=N) m ORDER BY resource_id,begin_time ASC","host":"10.*.*.206","log_timestamp":"1658657201","operate_type":"SELECT","node_id":"d285609201534696bdcd648519fe2b8dno02","instance_id":"5b67dc63ba824145aae1f12ff51e58b8in02"}

- 结构化字段及字段说明

表 4-95 结构化字段

字段	示例	描述	类型
log_type	slow_log	日志类型	string
execute_time	328.662	sql执行时间	float
user	authoring	用户	string
log_time	2022-07-24T10:06:41.000	日志打印时间	string
database	authoring-test	数据库名	string

字段	示例	描述	类型
statement	SELECT * FROM (SELECT n.user_id,n.id AS resource_id,e.create_at AS begin_time,e.create_at AS end_time ,N AS resource_spec_code,COALESCE(cast(e.flavor as varchar), cast(s.volume_size as varchar)) AS billing_unit,c.az_id,-N AS accumulate_factor_value,CONCAT(N, s.id, N) AS bss_params,n.project_id, n.domain_id, e.status , N AS resource_type , w.workspace_id,w.enterprise_project_id FROM t_resource_status_event e INNER JOIN t_notebook_evs_storage s on s.id=e.resource_id LEFT JOIN t_notebook_instance n on s.id=n.storage_id LEFT JOIN t_logic_cluster l on n.resource_cluster_id=l.id LEFT JOIN t_cce_cluster c on c.id=l.cce_id LEFT JOIN t_workspace w on w.workspace_id=n.workspace_id WHERE e.category = N AND s.resource_ownership=N AND e.create_at BETWEEN $N AND $N UNION ALL SELECT n.user_id,n.id AS resource_id,$N AS begin_time,$N AS end_time ,N AS resource_spec_code,COALESCE(cast(e.flavor as varchar), cast(s.volume_size as varchar)) AS billing_unit,c.az_id,-N AS accumulate_factor_value,CONCAT(N, s.id, N) AS bss_params,n.project_id, n.domain_id, e.status , N AS resource_type , w.workspace_id,w.enterprise_project_id FROM t_resource_status_event e INNER JOIN t_notebook_evs_storage s on s.id=e.resource_id LEFT JOIN t_notebook_instance n on s.id=n.storage_id LEFT JOIN t_logic_cluster l on n.resource_cluster_id=l.id LEFT JOIN t_cce_cluster c on c.id=l.cce_id LEFT JOIN t_workspace w on w.workspace_id=n.workspace_id INNER JOIN (SELECT	sql信息	string

字段	示例	描述	类型
	resource_id,max(create_at) as create_at FROM t_resource_status_event WHERE create_at < $N AND category = N GROUP BY resource_id) x ON e.resource_id=x.resource_id AND e.create_at=x.create_at WHERE e.create_at < $N AND e.category = N AND e.status = N AND s.resource_ownership=N) m ORDER BY resource_id,begin_time ASC		
host	10.*.*.206	ip	string
log_timestamp	1658657201	日志打印时间戳	string
operate_type	SELECT	操作类型	string
node_id	d285609201534696bdcd648519fe2b8dno02	节点id	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例id	string

POSTGRESQL 错误日志结构化模板日志详情

- POSTGRESQL错误日志示例

  表 4-96 结构化模板示例

模板名称	示例日志
POSTGRESQL错误日志	{"log_type":"error_log","severity":"WARNING","log_time":"2022-08-22T06:52:08Z","raw_message":"Occur error when reading bytes from a network handler. Client actively closes the connection.","node_id":"d285609201534696bdcd648519fe2b8dno02","instance_id":"5b67dc63ba824145aae1f12ff51e58b8in02"}

- 结构化字段及字段说明

  表 4-97 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	WARNING	日志级别	string
log_time	2022-08-22T06:52:08Z	日志产生时间	string

字段	示例	描述	类型
raw_message	Occur error when reading bytes from a network handler. Client actively closes the connection.	错误日志内容	string
node_id	d285609201534696bdcd648519fe2b8dno02	节点id	string
instance_id	5b67dc63ba824145aae1f12ff51e58b8in02	实例id	string

4.3.30 云数据库 RDS for SQLServer 接入 LTS

LTS支持云数据库（RDS）for SQLServer日志接入，具体接入方法请参见**日志管理**。

结构化模板日志详情

- SQLSERVER错误日志示例

表 4-98 结构化模板示例

模板名称	示例日志
SQLSERVER错误日志	{ "log_type": "error_log", "severity": "WARNING", "log_time": "2022-08-22T06:52:08Z", "raw_message": "Occur error when reading bytes from a network handler. Client actively closes the connection.", "node_id":"7346b0db609b463e976054928af50e85no01", "instance_id":"9c589b9d7a4d45dbaf7deb9f8520611cin01"}

- 结构化字段及字段说明

表 4-99 结构化字段

字段	示例	描述	类型
log_type	error_log	日志类型	string
severity	WARNING	日志级别	string
log_time	2022-08-22T06:52:08Z	日志产生时间	string
raw_message	Occur error when reading bytes from a network handler. Client actively closes the connection.	日志内容	string
node_id	7346b0db609b463e976054928af50e85no01	节点Id	string
instance_id	9c589b9d7a4d45dbaf7deb9f8520611cin01	实例Id	string

4.3.31 应用与数据集成平台 ROMA Connect 接入 LTS

LTS支持应用与数据集成平台（ROMA Connect）日志接入，具体接入方法请参见**查看API调用日志**。

4.3.32 视频直播 Live 接入 LTS

LTS支持视频直播服务（Live）日志接入，具体接入方法请参见**实时配置日志**。

4.3.33 消息通知服务 SMN 接入 LTS

LTS支持消息通知服务（SMN）日志接入，具体接入方法请参见**消息传输日志**。

结构化模板日志详情

- 日志示例

表 4-100 结构化模板示例

模板名称	示例日志
SMN	{"message_id":"1ae49922602a42fc83acb9689a2eb5f4","project_id":"5a9f32e4f1ec4bbe9695ff9da51c2925","topic_urn":"urn:smn:cn-north-1:5a9f32e4f1ec4bbe9695ff9da51c2925:demo","subscriber_urn":"urn:smn:cn-north-1:5a9f32e4f1ec4bbe9695ff9da51c2925:demo:b55c3c6fa7cd471b9f24818d530a8740","protocol_name":"https","endpoint":"https://127.0.0.1:443/https","status":"DELIVERED","http_code":200,"create_time":"2022-11-01T00:00:00Z","send_time":"2022-11-01T00:00:10Z"}

- 结构化字段及字段说明

表 4-101 结构化字段

字段	示例	描述	类型
message_id	1ae49922602a42fc83acb9689a2eb5f4	消息ID	string
project_id	5a9f32e4f1ec4bbe9695ff9da51c2925	项目ID	string
topic_urn	urn:smn:cn-north-1:5a9f32e4f1ec4bbe9695ff9da51c2925:demo	Topic的唯一的资源标识	string
subscriber_urn	urn:smn:cn-north-1:5a9f32e4f1ec4bbe9695ff9da51c2925:demo:b55c3c6fa7cd471b9f24818d530a8740	订阅者的唯一资源标识	string

字段	示例	描述	类型
protocol_name	https	不同协议对应不同的endpoint（接受消息的接入点）。目前支持的协议包括： “email”：邮件传输协议，endpoint为邮箱地址。 “sms”：短信传输协议，endpoint为手机号码。 “functiongraph” FunctionGraph（函数）传输协议，endpoint为一个函数 “functionstage”：。FunctionStage（工作流）传输协议，endpoint为一个函数工作流 “http”、“https”：HTTP/HTTPS传输协议，endpoint为URL。	string
endpoint	https://127.0.0.1:443/https	接受消息的接入点	string
status	DELIVERED	消息状态。目前包括以下状态： “DELIVERED”：已送达 “FAIL_DELIVERED”：发送失败 “REJECTS”：已拒绝。触发流控机制	string
http_code	200	HTTP返回码，仅支持HTTP/HTTPS协议消息	long
create_time	2022-11-01T00:00:00Z	消息创建时间。时间格式为UTC时间，YYYY-MM-DDTHH:MM:SSZ	string
send_time	2022-11-01T00:00:10Z	消息发送时间。时间格式为UTC时间，YYYY-MM-DDTHH:MM:SSZ	string

4.3.34 安全云脑 secMaster 接入 LTS

LTS支持安全云脑（SecMaster）日志接入，具体接入方法请参见**新增数据投递**。

4.3.35 虚拟私有云 VPC 接入 LTS

云日志服务支持虚拟私有云（Virtual Private Cloud）日志接入。

前提条件

购买并使用华为云VPC实例。

使用限制

目前支持的弹性云服务器，请参见**VPC流日志简介**。

设置 VPC 接入

云日志服务接入方式为虚拟私有云 VPC时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，单击“虚拟私有云 VPC”进行VPC接入配置。

步骤3 选择日志流。

1. 单击“所属日志组”后的目标框，在下拉列表中选择具体的日志组，若没有所需的日志组，单击“所属日志组”目标框后的“新建”，在弹出的创建日志组页面创建新的日志组。
2. 单击“所属日志流”后的目标框，在下拉列表中选择具体的日志流，若没有所需的日志流，单击“所属日志流”目标框后的“新建”，在弹出的创建日志流页面创建新的日志流。
3. 单击“下一步”：VPC配置。

步骤4 VPC配置。

单击“前往VPC配置”。

1. 在虚拟私有云控制台，选择“VPC流日志”。
2. 在VPC流日志页面中，单击“创建VPC流日志”，配置各参数信息。

   说明

   具体的操作步骤及参数配置，请见**虚拟私有云《用户指南》**。
3. 完成后，单击“确定”。

步骤5 单击下一步：日志流配置。

表 4-102 日志流配置参数表

参数	说明
自动对日志流进行结构化配置和索引配置	开启  该按钮，自动对日志流进行结构化配置和索引配置。日志流结构化配置为VPC系统模板；索引配置为所有VPC解析出来的字段打开快速分析按钮。配置结构化和索引后，才能对VPC日志进行SQL分析，并提供可视化图表。

参数	说明
自动为日志流添加标签：log_type=apig_layer_access	开启该按钮，自动为日志流添加标签（log_type=vpc_flow）后，VPC仪表盘模板才能匹配该日志流。
自动为日志流创建仪表盘	开启该按钮，自动为日志流创建VPC仪表盘。

步骤6 完成。

----结束

结构化模板日志详情

- VPC日志示例

  表 4-103 结构化模板示例

模板名称	示例日志
VPC	1 5f67944957444bd6bb4fe3b367de8f3d 1d515d18-1b36-47dc-a983-bd6512aed4bd 192.168.0.154 192.168.3.25 38929 53 17 1 96 1548752136 1548752736 ACCEPT OK

- 结构化字段及字段说明

  表 4-104 结构化字段

字段	示例	描述	类型
version	1	VPC流日志版本。	long
project_id	5f67944957444bd6bb4fe3b367de8f3d	项目ID。	string
interface_id	1d515d18-1b36-47dc-a983-bd6512aed4bd	为其记录流量的网卡的ID。	string
srcaddr	192.168.0.154	源地址。	string
dstaddr	192.168.3.25	目的地址。	string
srcport	38929	源端口。	long
dstport	53	目标端口。	long
protocol	17	IANA协议编号。有关更多信息，请参阅**Internet协议编号**。	long
packets	1	数据包的数量。	long
bytes	96	数据包的大小。	long

字段	示例	描述	类型
start	1548752136	捕获窗口启动的时间，采用Unix秒的格式。	long
end	1548752736	捕获窗口结束的时间，采用Unix秒的格式。	long
action	ACCEPT	与流量关联的操作： • ACCEPT：安全组或网络ACL允许记录的流量。 • REJECT：安全组或者网络ACL拒绝记录的流量。	string
log_status	OK	流日志的日志记录状态： • OK：数据正常记录到选定目标。 • NODATA：捕获窗口中没有传入或传出符合“采集类型”的网卡的网络流量。 • SKIPDATA：捕获窗口中跳过了一些流日志记录。这可能是由于内部容量限制或内部错误。 示例： 如果您创建VPC流日志时设置“采集类型”为“接受”，当有接受流量时，“log-status”将显示为“OK”。当没有接受的流量时，不管是否有拒绝的流量，“log-status”都将显示为“NODATA”。当有一些接受流量异常跳过时，“log-status”将显示为“SKIPDATA”。	string

4.3.36 Web 应用防火墙 WAF 接入 LTS

LTS支持Web应用防火墙 WAF接入，具体接入方法请参见**开启全量日志**。

WAF 访问结构化模板日志详情

- WAF访问日志示例

表 4-105 结构化模板示例

模板名称	示例日志
WAF访问日志	{"response_code":"504","scheme":"http","upstream_addr":"100.93.2.229:80","body_bytes_sent":"163","upstream_header_time":"-","connection_requests":"1","ssl_cipher":"","hostid":"1736cc7331b74b198e2ef07555a970ce","pid":"2152","tls_version":"","http_host":"www.testh.com","process_time":"0","access_stream_id":"88003425-d7bc-46ce-8ae7-77a8aa18a814","time_iso8601":"2022-07-29T19:39:10+08:00","intel_crawler":"","upstream_status":"504","remote_ip":"10.63.46.110","request_time":"30.008","tenantid":"1d26cc8c86a840e28a4f8d0d07852f1d","sip":"10.63.46.110","bytes_send":"420","projectid":"2a473356cca5487f8373be891bffc1cf","user_agent":"curl/7.29.0","web_tag":"","method":"GET","bind_ip":"10.63.36.208","region_id":"","remote_port":"20582","ssl_ciphers_md5":"","x_real_ip":"","url":"/","x_forwarded_for":"","sni":"","args":"public/../style/general.css=true","cdn_src_ip":"","enterprise_project_id":"0","upstream_connect_time":"-","engine_id":"","request_length":"110","group_id":"5d574e6a-87da-42bc-bfd4-ff61a1b336a4","requestid":"36f0a9212b14528ffc090f1811cd87d8","ssl_curves":"","ssl_session_reused":"","waf-time":"2022-07-29T11:39:10.000Z","upstream_response_time":"30.008","time":"29/Jul/2022:19:39:10 +0800","category":"access","eng_ip":"10.63.36.208"}

- 结构化字段及字段说明

表 4-106 结构化字段

字段	示例	描述	类型
response_code	504	源站返回给WAF的响应状态码。	string
scheme	http	请求所使用的协议有： • http • https	string
upstream_addr	100.93.2.229:80	选择的后端服务器地址。例如，WAF回源到ECS，则返回源站ECS的IP。	string
body_bytes_sent	163	发送给客户端的响应体字节数	string
upstream_header_time	-	后端服务器接收到第一个响应头字节的用时	string
connection_requests	1	连接请求	string
ssl_cipher	-	SSL密码	string
hostid	1736cc7331b74b198e2ef07555a970ce	访问请求的域名标识	string
pid	2152	*进程ID*	string

字段	示例	描述	类型
tls_version	-	建立SSL连接的协议版本	string
http_host	www.testh.com	请求的服务器域名	string
process_time	0	引擎的检测用时	string
access_stream_id	88003425-d7bc-46ce-8ae7-77a8aa18a814	日志流ID	string
time_iso8601	2022-07-29T19:39:10+08:00	日志的ISO 8601格式时间	string
intel_crawler	-	爬虫	string
upstream_status	504	后端服务器的响应码	string
remote_ip	10.63.46.110	请求的客户端IP	string
request_time	30.008	请求处理时间	string
tenantid	1d26cc8c86a840e28a4f8d0d07852f1d	防护域名的租户ID	string
sip	10.63.46.110	客户端请求IP	string
bytes_send	420	发送给客户端的总字节数	string
projectid	2a473356cca5487f8373be891bffc1cf	防护域名的项目ID	string
user_agent	curl/7.29.0	请求header中的 user-agent	string
web_tag	-	网站名称	string
method	GET	请求方法	string
bind_ip	10.63.36.208	WAF引擎回源IP	string
region_id	-	请求所属区域	string
remote_port	20582	远程端口	string
ssl_ciphers_md5	-	ssl_ciphers 的 md5 值	string

字段	示例	描述	类型
x_real_ip	-	当WAF前部署代理时，真实的客户端IP	string
url	/	请求URL	string
x_forwarded_for	-	请求头中x_forwarded_for的内容	string
sni	-	通过SNI请求的域名	string
args	public/../style/general.css=true	URL 中的参数数据	string
cdn_src_ip	-	当WAF前部署CDN时CDN识别到的客户端IP	string
enterprise_project_id	0	请求域名所属企业项目ID	string
upstream_connect_time	-	后端服务器连接用时	string
engine_id	-	WAF引擎标识	string
request_length	110	请求的长度	string
group_id	5d574e6a-87da-42bc-bfd4-ff61a1b336a4	对接LTS服务的日志组ID	string
requestid	36f0a9212b14528ffc090f1811cd87d8	随机ID标识	string
ssl_curves	-	客户端支持的曲线列表	string
ssl_session_reused	-	SSL会话是否被重用	string
waf-time	2022-07-29T11:39:10.000Z	WAF日志时间	string
upstream_response_time	30.008	后端服务器响应时间	string
time	29/Jul/2022:19:39:10 +0800	访问请求的时间	string

字段	示例	描述	类型
waf_category	access	WAF日志类别	string
eng_ip	10.63.36.208	WAF引擎IP	string

WAF 攻击结构化模板日志详情

- WAF攻击日志示例

表 4-107 结构化模板示例

模板名称	示例日志
WAF攻击日志	{"policy_id":"cd081ba3d6674000acc37d7e2a4b9140","hport":"80","body_bytes_sent":"163","hostid":"1736cc7331b74b198e2ef07555a970ce","rule":"040002","engine_ip":"10.63.36.208","pid":"2152","http_host":"www.testh.com","process_time":"1","reqid":"0000-0000-0000-20820220729193940-f34cf25e","time_iso8601":"2022-07-29T19:39:40+08:00","upstream_status":"504","hit_data":"public/../style/general.css","attack_stream_id":"98de5d5a-9f54-4d01-9882-eca7bec99d09","remote_ip":"10.63.46.110","attack":"lfi","tenantid":"1d26cc8c86a840e28a4f8d0d07852f1d","host":"www.testh.com","action":"log","backend":{"protocol":"HTTP","alive":true,"port":80,"host":"100.93.2.229","weight":1,"type":"ip"},"id":"04-0000-0000-0000-20820220729193940-f34cf25e","sip":"10.63.46.110","projectid":"2a473356cca5487f8373be891bfc1cf","web_tag":"","attack-time":"2022-07-29T11:39:40.000Z","method":"GET","cookie":"{\"HWWAFSESTIME\":\"1659094780939\",\"HWWAFSESID\":\"e2cd0733b4712e4cc4\"}","level":2,"params":"{\"public\/..\/style\/general.css\":\"true\"}","x_real_ip":"","uri":"/","x_forwarded_for":"","cdn_src_ip":"","enterprise_project_id":"0","req_body":"","engine_id":"","group_id":"5d574e6a-87da-42bc-bfd4-ff61a1b336a4","requestid":"f34cf25eb33ed82cd7261a8276a60c39","multipart":"null","header":"{\"host\":\"www.testh.com\",\"user-agent\":\"curl\/7.29.0\",\"accept\":\"*\/*\"}","location":"params","upstream_response_time":"30.000","time":"2022-07-29 19:39:40","category":"attack","sport":28408,"status":"504"}

- 结构化字段及字段说明

表 4-108 结构化字段

字段	示例	描述	类型
policy_id	cd081ba3d6674000acc37d7e2a4b9140	防护策略ID	string
hport	80	请求的服务器端口	string
body_bytes_sent	163	发送给客户端的响应体字节数	string
hostid	1736cc7331b74b198e2ef07555a970ce	防护域名ID（upstream_id）	string

字段	示例	描述	类型
rule	040002	触发的规则ID或者自定义的策略类型描述	string
engine_ip	10.63.36.208	引擎 IP	string
pid	2152	进程ID	string
http_host	www.testh.com	请求的服务器域名	string
process_time	1	引擎的检测用时	string
reqid	0000-0000-0000-20820220729193940-f34cf25e	随机ID标识	string
time_iso8601	2022-07-29T19:39:40+08:00	日志的ISO 8601格式时间	string
upstream_status	504	后端服务器的响应码	string
hit_data	public/../style/general.css	触发恶意负载的字符串	string
attack_stream_id	98de5d5a-9f54-4d01-9882-eca7bec99d09	日志流ID	string
remote_ip	10.63.46.110	请求的客户端IP	string

字段	示例	描述	类型
attack	lfi	发生攻击的类型，仅在攻击日志中出现。 • default：默认 • sqli：SQL注入攻击 • xss：跨站脚本攻击 • webshell：WebShell攻击 • robot：恶意爬虫 • cmdi：命令注入攻击 • rfi：远程文件包含 • lfi： 本地文件包含 • illegal：非法请求 • vuln：漏洞攻击 • cc：命中CC防护规则 • custom_custom：命中精准防护规则 • custom_whiteip：命中IP黑白名单规则 • custom_geoip：命中地理位置控制规则 • antitamper： 命中网页防篡改规则 • anticrawler：命中JS挑战反爬虫规则 • leakage：命中敏感信息泄露规则 • followed_action：攻击惩罚，详见**配置攻击惩罚标准**。	string
tenantid	1d26cc8c86a840e28a4f8d0d07852f1d	防护域名的租户ID	string
host	www.testh.com	请求的服务器域名	string
action	log	WAF防护攻击动作。 • block：拦截 • log：仅记录 • captcha：人机验证	string
backend.protocol	HTTP	当前后端协议	string
backend.alive	true	当前后端状态	string

字段	示例	描述	类型
backend.port	80	当前后端端口	long
backend.host	100.93.2.229	当前后端Host值	string
backend.weight	1	当前后端权重	long
backend.type	ip	当前后端Host类型	string
id	04-0000-0000-0000-20820220729193940-f34cf25e	请求ID标识	string
sip	10.63.46.110	请求的客户端IP	string
projectid	2a473356cca5487f8373be891bffc1cf	防护域名的项目ID	string
web_tag	-	网站名称	string
attack-time	2022-07-29T11:39:40.000Z	攻击时间	string
method	GET	请求方法	string
cookie	{"HWWAFSESTIME":"1659094780939","HWWAFSESID":"e2cd0733b4712e4cc4"}	Cookie内容	string
level	2	表示Web基础防护策略级别。 ● 1：宽松 ● 2：中等 ● 3：严格	long
params	{"public\/..\/style\/general.css":"true"}	请求URI后的参数信息	string
x_real_ip	-	当WAF前部署代理时，真实的客户端IP	string
uri	/	请求URI	string
x_forwarded_for	-	请求头中x_forwarded_for的内容	string
cdn_src_ip	-	当WAF前部署CDN时CDN识别到的客户端IP	string
enterprise_project_id	0	请求域名所属企业项目ID	string
req_body	-	请求体	string

字段	示例	描述	类型
engine_id	-	WAF引擎标识	string
group_id	5d574e6a-87da-42bc-bfd4-ff61a1b336a4	group_id	string
requestid	f34cf25eb33ed82cd7261a8276a60c39	随机ID标识	string
multipart	null	multipart	string
header	{"host":"www.testh.com","user-agent":"curl\/7.29.0","accept":"*\/*"}	请求header信息	string
location	params	触发恶意负载的位置	string
upstream_response_time	30.000	后端服务器响应时间	string
time	2022-07-29 19:39:40	日志时间	string
waf_category	attack	WAF日志类别	string
sport	28408	客户端请求端口	long
status	504	请求的响应状态码	string

4.4 使用 API 接入 LTS

4.4.1 API 接入概述

您可以通过调用云日志服务（LTS）提供REST风格的API将日志上报到LTS，具体有上报日志和上报高精度日志两个接口。

以下是两个接口的适用场景和各区域访问IP：

表 4-109 适用场景

API名称	日志时间	举例说明	适用场景

上报日志	用户调用API上传一批日志时，通过log_time_ns字段指定一个初始时间。
	每一条日志的日志时间，使用log_time_ns+顺序计数得到。
		{
		"log_time_ns": "1586850540000000000",
		"contents": [
		"log1",
		"log2"
		],
		"labels": {
		"user_tag": "string"
		}
		}
		上报到LTS后：
		log1的日志时间为：**1586850540000000000**
		log2的日志时间为：**1586850540000000001**
			上传的一批日志是在相近时间点、按顺序产生的。

上报高精度日志	用户调用API上传一批日志时，每一条日志都需要通过log_time_ns字段指定日志时间。	{
		"contents":[
		{
		"log_time_ns":"1586850540000000000",
		"log":"log3"
		},
		{
		"log_time_ns":"1586850540000000008",
		"log":"log4"
		}
		],
		"labels":{
		"user_tag":"string"
		}
		}
		上报到LTS后：
		log3的日志时间为：**1586850540000000000**
		log4的日志时间为：**1586850540000000008**
			上传的一批日志是在不同时间、非顺序产生的，希望每条日志的时间单独指定。

表 4-110 访问 IP（accessip）

区域	访问IP
华北-北京一	100.125.57.101
华北-北京二	100.125.6.108

区域	访问IP
华北-北京四	100.125.12.150
华东-上海一	100.125.11.177
华东-上海二	100.125.140.102
华南-广州	100.125.158.115
西南-贵阳一	100.125.0.27
华南-广州-友好用户环境	100.125.4.30
亚太-新加坡	100.125.4.58

说明

您可以在云日志服务控制台，左侧导航栏中选择“主机管理”，单击“安装ICAgent”，在安装ICAgent页面中的命令里获取访问IP（accessip）。

4.4.2 上报日志接口参考

功能介绍

该接口用于主机上报租户日志给LTS。

接入点IP可在LTS控制台“主机管理 > 安装ICAgent”的安装命令中获取，端口为8102，调用时使用该参数请参见“**请求示例**”。

URI

POST /v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents

表 4-111 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：32 最大长度：32

参数	是否必选	参数类型	描述
log_group_id	是	String	日志组ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：36 最大长度：36
log_stream_id	是	String	日志流ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：36 最大长度：36 **注：每个logstream写入速率最大不能超过100MB/S，超过此规格可能会导致日志丢失。**

请求参数

表 4-112 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	从IAM获取的用户Token，获取方式请参见：**获取用户Token**。 缺省值：None 最小长度：1000 最大长度：2000
Content-Type	是	String	该字段填为：application/json;charset=UTF-8。 缺省值：None 最小长度：30 最大长度：30

表 4-113 请求 Body 参数

参数	是否必选	参数类型	描述
log_time_ns	是	Long	日志数据上报时间，UTC时间（纳秒）。 **说明** 通过接口上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。
contents	是	Array of String	日志内容。
labels	是	Object	用户自定义label。 **说明** 请不要将字段名称设置为**内置保留字段**，否则可能会造成字段名称重复、查询不精确等问题。
tenant_project_id	否	String	租户ID。

响应参数

状态码为 200 时:

表 4-114 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值： • SVCSTG.ALS.200.200
errorMessage	String	调用失败响应信息描述。 枚举值： • Report success.
result	String	响应结果。

状态码为 400 时:

表 4-115 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值： ● SVCSTG.ALS.200.201 ● SVCSTG.ALS.200.210
errorMessage	String	调用失败响应信息描述。 枚举值： ● Request conditions must be json format. ● projectid xxx log's quota has full!!
result	String	响应结果。

状态码为 401 时:

表 4-116 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值： ● SVCSTG.ALS.403.105
errorMessage	String	调用失败响应信息描述。 枚举值： ● Project id is invalid.
result	String	响应结果。

状态码为 500 时:

表 4-117 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值： ● LTS.200500
errorMessage	String	调用失败响应信息描述。 枚举值： ● Internal error

参数	参数类型	描述
result	String	响应结果。

状态码为 503 时:

表 4-118 响应 Body 参数

参数	参数类型	描述
result	String	ServiceUnavailable。 被请求的服务无效, 服务不可用。

请求示例

```
POST https://{接入点IP:8102}/v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents

{
    "log_time_ns": "1586850540000000000",
    "contents": [
        "Fri Feb  1 07:48:04 UTC 2019 0\n",
        "Sat Apri 18 16:04:04 UTC 2019"
    ],
    "labels": {
        "user_tag": "string"
    }
}
```

响应示例

状态码： 200

日志上报成功。

```
{
  "errorCode": "SVCSTG.ALS.200.200",
  "errorMessage": "Report success.",
  "result": null
}
```

状态码： 401

在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。

```
{
  "errorCode" : "SVCSTG.ALS.403.105",
  "errorMessage" : "Project id is invalid.",
  "result": null
}
```


状态码

状态码	描述
200	请求响应成功。
400	BadRequest。非法请求。建议根据error_msg直接修改该请求，不要重试该请求。
401	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
500	系统内部错误。
503	ServiceUnavailable。被请求的服务无效，服务不可用。

4.4.3 上报高精度日志接口参考

功能介绍

该接口用于主机上报租户日志给LTS。

接入点IP可在LTS控制台“主机管理 > 安装ICAgent”的安装命令中获取，端口为8102，调用时使用该参数请参见“**请求示例**”。

说明

每次上报的时候， 每条日志都必须带一个纳秒级的时间戳。在LTS界面查看日志的时候， 会按照时间戳排序展示在页面上。

URI

POST /v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents/high-accuracy

表 4-119 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：32 最大长度：32
log_group_id	是	String	日志组ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：36 最大长度：36

参数	是否必选	参数类型	描述
log_stream_id	是	String	日志流ID，获取方式请参见：**获取账号ID、项目ID、日志组ID、日志流ID**。 缺省值：None 最小长度：36 最大长度：36 **注：每个logstream写入速率最大不能超过100MB/S，超过此规格可能会导致日志丢失。**

请求参数

表 4-120 请求 Header 参数

参数	是否必选	参数类型	描述
X-Auth-Token	是	String	从IAM获取的用户Token，获取方式请参见：**获取用户Token**。 缺省值：None 最小长度：1000 最大长度：2000
Content-Type	是	String	该字段填为：application/json;charset=UTF-8。 缺省值：None 最小长度：30 最大长度：30
Content-Encoding	否	String	日志压缩格式 枚举值： • **GZIP** • **SNAPPY** • **gzip** • **snappy**

表 4-121 请求 Body 参数

参数	是否必选	参数类型	描述
contents	是	Array of **LogContents**	包含上报时间戳的日志内容列表。
labels	是	Object	用户自定义label。 **说明** 请不要将字段名称设置为**内置保留字段**，否则可能会造成字段名称重复、查询不精确等问题。
tenant_project_id	否	String	租户ID。

表 4-122 LogContents

参数	是否必选	参数类型	描述
log_time_ns	是	Long	日志数据上报时间，UTC时间（纳秒）。 **说明** 通过接口上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。
log	是	String	日志内容。

响应参数

状态码为 200 时:

表 4-123 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值： • SVCSTG.ALS.200.200
errorMessage	String	调用失败响应信息描述。 枚举值： • Report success.
result	String	响应结果。

状态码为 400 时:

表 4-124 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值: ● SVCSTG.ALS.200.201 ● SVCSTG.ALS.200.210
errorMessage	String	调用失败响应信息描述。 枚举值: ● Request conditions must be json format. ● projectid xxx log's quota has full!!
result	String	响应结果。

状态码为 401 时:

表 4-125 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值: ● SVCSTG.ALS.403.105
errorMessage	String	调用失败响应信息描述。 枚举值: ● Project id is invalid.
result	String	响应结果。

状态码为 500 时:

表 4-126 响应 Body 参数

参数	参数类型	描述
errorCode	String	错误码。 枚举值: ● LTS.200500

参数	参数类型	描述
errorMessage	String	调用失败响应信息描述。 枚举值： ● Internal error
result	String	响应结果。

状态码为 503 时:

表 4-127 响应 Body 参数

参数	参数类型	描述
result	String	ServiceUnavailable。 被请求的服务无效, 服务不可用。

请求示例

```
POST https://{接入点IP:8102}/v2/{project_id}/lts/groups/{log_group_id}/streams/{log_stream_id}/tenant/contents/high-accuracy

{
   "contents": [
      {
         "log_time_ns": "1586850540000000000",
         "log": "Fri Feb  15 15:48:04 UTC 2019"
      },
      {
         "log_time_ns": "1586850540000000001",
         "log": "Sat Apri 18 16:04:04 UTC 2019"
      }
   ],
   "labels": {
      "user_tag": "string"
   }
}
```

响应示例

状态码： 200

日志上报成功。

```
{
  "errorCode": "SVCSTG.ALS.200.200",
  "errorMessage": "Report success.",
  "result": null
}
```

状态码： 401

在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。

```
{
  "errorCode" : "SVCSTG.ALS.403.105",
```



```
  "errorMessage" : "Project id is invalid.",
  "result": null
}
```

状态码

状态码	描述
200	请求响应成功。
400	BadRequest。非法请求。建议根据error_msg直接修改该请求，不要重试该请求。
401	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
500	系统内部错误。
503	ServiceUnavailable。被请求的服务无效，服务不可用。

4.5 使用 SDK 接入 LTS

4.5.1 云日志服务 Java SDK

云日志服务SDK提供了Java语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（ project id ），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 已安装Java开发环境。日志服务Java SDK支持JDK1.8及以上，您可以执行 *java –version* 命令检查您已安装的Java版本。如未安装，可以从**Java官方网站下载安装包**进行安装。

使用说明

- 当用户修改权限后，权限信息在一天后生效。
- SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。使用详情见**Appender配置参数说明表**中的“enableLocalTest”参数，当该参数为true时，上报日志规格为单个机器200次/

秒（即每秒只能发送200次，每次批量发送数量/大小详情见参数“batchSizeThresholdInBytes、batchCountThreshold、lingerMs”）。

- 通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

注意事项

由于Java-SDK默认对SK明文存储，不符合某些对于安全有更高要求的用户，lts提供了一种用户自定义的加解密方式，建议用户优先使用加解密方式。

加密过程如下：

1. 用户编写一个java类，比如com.demo.DecryptDemo类，在该类中增加一个解密方法，比如decrypt方法，输入和输出均为字符串。
2. 编写decrypt的方法内容，客户自行实现SK加解密算法，返回解密后的值。
3. 当用户调用Java-SDK初始化时，会需要传入SK，这时使用DecryptDemo.decrypt方法即可。

安装 SDK

您可以通过以下两种方式安装日志服务Java SDK。

方式一：在Maven项目中加入依赖项（推荐方式）。

1. maven构建时，settings.xml文件的mirrors节点需要增加**华为外部开源仓**。如下：
```
<mirror>
    <id>huaweicloud</id>
    <mirrorOf>*</mirrorOf>
    <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```
2. 您可以在**华为外部开源仓**中获取日志服务Java SDK依赖的最新版本。
3. 在Maven工程中使用日志服务Java SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。以1.0.1版本为例，在<dependencies>中加入如下内容：
```
<dependency>
    <groupId>io.github.huaweicloud</groupId>
    <artifactId>lts-sdk-common</artifactId>
    <version>1.0.1</version>
</dependency>
<dependency>
    <groupId>io.github.huaweicloud</groupId>
    <artifactId>lts-sdk-java</artifactId>
    <version>1.0.1</version>
</dependency>
```

方式二：在项目中直接依赖Java SDK的jar包。

1. 下载**lts-sdk-common**和**lts-sdk-java**包。（建议使用最新版本的jar包。）
2. 请确保您的项目中有以下依赖，因为SDK的jar包中需要该依赖。
```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.83</version>
</dependency>
<dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.10.14</version>
</dependency>
```



```
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.11</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>30.1.1-jre</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
</dependency>
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>2.0.7</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.7</version>
</dependency>
```

3. 以0.6.75版本的IntelliJ IDEA为例：项目中导入JAR包
   - 在IntelliJ IDEA中选择您的工程，选择File > Project Structure。
   - 在Project Structure对话框，左侧单击Modules。
   - 在Dependencies页签，选择+ > JARs or directories。
   - 在Attach Files or Directories对话框中，选中已下载的JAR文件，单击OK。

示例代码

```
package com.huawei.log;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.alibaba.fastjson.JSONObject;

import com.huaweicloud.lts.appender.JavaSDKAppender;
import com.huaweicloud.lts.producer.Producer;
import com.huaweicloud.lts.producer.exception.ProducerException;
import com.huaweicloud.lts.producer.model.log.LogContent;
import com.huaweicloud.lts.producer.model.log.LogItem;


public class LtsLogJavaSdkTest1 implements Runnable {

    @Override
    public void run() {
        /* 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密文存放, 使用时解密, 确保安全;
        本示例以ak和sk保存在环境变量中为例, 运行本示例前请先在本地环境中设置环境变量HUAWEICLOUD_SDK_AK和HUAWEICLOUD_SDK_SK. */
        String ak = System.getenv("HUAWEICLOUD_SDK_AK");
```



```
        String sk = System.getenv("HUAWEICLOUD_SDK_SK");
        // 构建appender
        JavaSDKAppender appender = JavaSDKAppender.custom()
        // 华为云账号的项目ID（project id）
        .setProjectId("xxx")
        // 华为云账号的AK
        .setAccessKeyId(ak)
        // 华为云账号的SK
        .setAccessKeySecret(sk)
        // 云日志服务的区域
        .setRegionName("xxxxxx")
        // 单个Appender能缓存的日志大小上限
        .setTotalSizeInBytes(104857600)
        // producer发送日志时阻塞时间
        .setMaxBlockMs(0L)
        // 执行日志发送任务的线程池大小
        .setIoThreadCount(8)
        // producer发送单批日志量上限
        .setBatchSizeThresholdInBytes(524288)
        // producer发送单批日志条数上限
        .setBatchCountThreshold(4096)
        // producer发送单批日志等待时间
        .setLingerMs(2000)
        // producer发送日志失败后重试次数
        .setRetries(5)
        // 首次重试的退避时间
        .setBaseRetryBackoffMs(500L)
        // 重试的最大退避时间
        .setMaxRetryBackoffMs(500L)
       // 默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志
       // .setEnableLocalTest(false)
        // 超过1M的日志, 拆分后丢弃大于1M的数据
        // .setGiveUpExtraLongSingleLog(true)
        .builder();
    // 获取发送producer
    Producer producer = appender.getProducer();

    // 创建日志发送的结构体
    while (true) {
        List<LogItem> logItemList = new ArrayList<>();
        try {
            Thread.sleep(500);
            // 构建日志
            LogItem logItem = getLogItem();
            logItemList.add(logItem);
            // 发送日志（日志组ID，日志流ID，日志）
            producer.send("xxxxxx", "xxxxxx", logItemList);
        } catch (InterruptedException | ProducerException e) {
            e.printStackTrace();
        }
    }
}

private LogItem getLogItem() {
    LogItem logItem = new LogItem();
    // 华为云账号的项目ID（project id）
    logItem.setTenantProjectId("xxxxxx");
    // 客户可以自行定义日志标签, 如果不需要请传入一个空Map
    Map<String, String> labels = new HashMap<>();
    labels.put("javaSdk1", "value1");
    labels.put("javaSdk2", "value2");
    logItem.setLabels(JSONObject.toJSONString(labels));
```

```
        List<LogContent> contents = new ArrayList<>();

        for (int k = 0; k < 100; k++) {
            LogContent logContent = new LogContent();
            // 日志产生时间, 建议为当前时间Ns
            long logTimeNs = System.currentTimeMillis() * 1000000L + System.nanoTime() %
1000000L;
            logContent.setLogTimeNs(logTimeNs);
            // 日志内容
            logContent.setLog("test~~javaSdk~~~test1 " + k);
            contents.add(logContent);
        }
        logItem.setContents(contents);
        return logItem;
    }

}
```

配置参数说明

- Appender中配置参数说明

表 4-128 Appender 配置参数说明表

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
regionName	云日志服务的区域。	String	必填	-
logGroupId	LTS的日志组ID。	String	必填	-
logStreamId	LTS的日志流ID。	String	必填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	100M（100 * 1024 * 1024）

参数名称	描述	类型	是否需要填写	默认值
maxBlockMs	如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，单位为毫秒。默认为 60 秒（60000毫秒），建议为0秒。 ● 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 ● 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60秒（60000毫秒）
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	0.5M（512 * 1024）
batchCountThreshold	当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送。	int	选填	4096
lingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2S
retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 5 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1S
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50S
giveUpExtraLongSingleLog	超过1M的日志, 拆分后丢弃大于1M的数据。	boolean	选填	false

参数名称	描述	类型	是否需要填写	默认值
enableLocalTest	是否开启跨云上报日志。 ● 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问（建议调测使用）。 ● 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false

- 日志上报LogItem类参数说明

表 4-129 日志上报 LogItem 类参数说明表

参数名称	描述	类型	是否需要填写
tenantProjectId	华为云账号的项目ID（project id）。	String	是
contents	批量日志内容。	Array<LogContent>	是

labels	日志标签，JSON格式。
	支持的Key值有：projectId，clusterId，clusterName，nameSpace，appGroup，appName，serviceID，podName，pid，group，containerName，hostId，hostIP，hostName，hostIPv6，serverlessFunc，serverlessPkg，functionVersion，pailId，pathFile，kind
	例如：
	{
	"clusterName" : "k8s",
	"appGroup" : "group1"
	}
		String	是

说明

单次上报条数小于4096条，且小于5M。

- 日志上报LogConten类参数说明

参数名称	描述	类型	是否需要填写
logTimeNs	日志时间（单位：纳秒）	long	是
log	日志内容	String	是

说明

通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京二	cn-north-2
华北-北京四	cn-north-4
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.2 云日志服务 Java SDK（log4j2）

云日志服务Java SDK提供了与log4j2适配的扩展插件，可以直接在log4j2中配置华为云appender，将通过log4j2产生的日志直接上报至云日志服务。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为账号并开通华为云。

- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 已安装Java开发环境。日志服务Java SDK支持JDK1.8及以上，您可以执行 *java -version* 命令检查您已安装的Java版本。如未安装，可以从**Java官方网站下载安装包**进行安装。

使用说明

- 当用户修改权限后，权限信息在一天后生效。
- SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。使用详情见**Appender配置参数说明表**中的“enableLocalTest”参数，当该参数为true时，上报日志规格为单个机器200次/秒（即每秒只能发送200次，每次批量发送数量/大小详情见参数“batchSizeThresholdInBytes、batchCountThreshold、lingerMs”）。
- 通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

安装 SDK

您可以通过以下两种方式安装日志服务log4j2 SDK

方式一：在Maven项目中加入依赖项（推荐方式）。

1. maven构建时，settings.xml文件的mirrors节点需要增加**华为外部开源仓**。如下：
```
<mirror>
    <id>huaweicloud</id>
    <mirrorOf>*</mirrorOf>
    <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```
2. 您可以在**华为外部开源仓**中获取日志服务log4j2 SDK依赖的最新版本。
3. 在Maven工程中使用日志服务log4j2 SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。以1.0.1版本为例，在<dependencies>中加入如下内容：
```
<dependency>
    <groupId>io.github.huaweicloud</groupId>
    <artifactId>lts-sdk-common</artifactId>
    <version>1.0.1</version>
</dependency>
<dependency>
    <groupId>io.github.huaweicloud</groupId>
    <artifactId>lts-sdk-log4j2</artifactId>
    <version>1.0.1</version>
</dependency>
```

方式二：在项目中直接依赖Log4j2 SDK的jar包。

1. 下载**lts-sdk-common**和**lts-sdk-log4j2**包。
2. 请确保您的项目中有以下依赖，因为SDK的jar包中需要该依赖。
```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
```



```
    <version>1.2.83</version>
</dependency>
<dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.10.14</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.11</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>30.1.1-jre</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
</dependency>
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>2.0.7</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.7</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.19.0</version>
</dependency>
<dependency>
   <groupId>org.apache.logging.log4j</groupId>
   <artifactId>log4j-api</artifactId>
   <version>2.19.0</version>
</dependency>
```

3. 以0.6.75版本的IntelliJ IDEA为例：项目中导入JAR包
   - 在IntelliJ IDEA中选择您的工程，选择File > Project Structure。
   - 在Project Structure对话框，左侧单击Modules。
   - 在Dependencies页签，选择+ > JARs or directories。
   - 在Attach Files or Directories对话框中，选中已下载的JAR文件，单击OK。

示例代码

1. log4j2.xml示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
 <Configuration status="WARN">
```



```
    <Appenders>
        <!-- projectId：华为云账号的项目ID-->
        <!-- logGroupId：华为云日志服务日志组ID-->
        <!-- logStreamId：华为云日志服务日志流ID-->
        <!-- accessKeyId：华为云账号的AK, 认证用的AK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放, 使用时解密, 确保安全-->
        <!-- accessKeySecret：华为云账号的SK, 认证用的SK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放, 使用时解密, 确保安全-->
        <!-- regionName：云日志服务的区域：单个Appender能缓存的日志大小上限-->
        <!-- maxBlockMs：producer发送日志时阻塞时间-->
        <!-- ioThreadCount：执行日志发送任务的线程池大小-->
        <!-- batchSizeThresholdInBytes：producer发送单批日志量上限-->
        <!-- batchCountThreshold：producer发送单批日志条数上限-->
        <!-- lingerMs：producer发送单批日志等待时间-->
        <!-- retries：producer发送日志失败后重试次数-->
        <!-- baseRetryBackoffMs：首次重试的退避时间-->
        <!-- maxRetryBackoffMs：重试的最大退避时间-->
        <!-- giveUpExtraLongSingleLog：过1M的日志, 拆分后放弃1M以外的-->
        <!-- enableLocalTest：默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志-->
        <ltsLog name="log4j2sdkappender"
                projectId="xxx"
                logGroupId="xxx"
                logStreamId="xxx"
                accessKeyId="xxx"
                accessKeySecret="xxx"
                regionName="xxx"
                totalSizeInBytes="104857600"
                maxBlockMs="0"
                ioThreadCount="8"
                batchSizeThresholdInBytes="524288"
                batchCountThreshold="4096"
                lingerMs="2000"
                retries="5"
                baseRetryBackoffMs="500"
                maxRetryBackoffMs="500"
                giveUpExtraLongSingleLog="true"
                enableLocalTest="false">
            <PatternLayout pattern="%d %-5level [%thread] %logger{0}: %msg"/>
        </ltsLog>
    </Appenders>

    <Loggers>
        <Root level="WARN">
            <AppenderRef ref="log4j2sdkappender"/>
        </Root>
    </Loggers>

</Configuration>
```

2. 编写代码产生的日志如下：

```
package com.huawei.log;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

public class TestLog4j2AppenderInfo {

    private static final Logger LOGGER = LogManager.getLogger(TestLog4j2AppenderInfo.class);

    public static void main(String[] args) {
        LOGGER.info("test~log4j2~~~info");
    }
```



```
}
```

配置参数说明

- Appender配置参数说明

表 4-130 Appender 配置参数说明

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（ project id ）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
regionName	云日志服务的区域。	String	必填	-
logGroupId	LTS的日志组ID。	String	必填	-
logStreamId	LTS的日志流ID。	String	必填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	100M（100 * 1024 * 1024）
maxBlockMs	如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，单位为毫秒。默认为 60 秒（60000 毫秒），建议为0秒。 ● 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 ● 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60秒（60000毫秒）

参数名称	描述	类型	是否需要填写	默认值
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	0.5M（512 * 1024）
batchCountThreshold	当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送。	int	选填	4096
lingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2S
retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 5 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1S
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50S
giveUpExtraLongSingleLog	超过1M的日志, 拆分后丢弃大于1M的数据。	boolean	选填	false
enableLocalTest	是否开启跨云上报日志。 ● 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问（建议调测使用）。 ● 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false

参数获取方法

- 区域表

表 4-131 区域表

区域名称	RegionName
华北-北京二	cn-north-2
华北-北京四	cn-north-4
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.3 云日志服务 LogBack SDK

云日志服务Java SDK提供了与logback适配的扩展插件，可以直接在logback中配置华为云appender，将通过logback产生的日志直接上报至云日志服务。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 已安装Java开发环境。日志服务Java SDK支持JDK1.8及以上，您可以执行 *java -version* 命令检查您已安装的Java版本。如未安装，可以从**Java官方网站下载安装包**进行安装。

使用说明

- 当用户修改权限后，权限信息在一天后生效。
- SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一。使用详情见**Appender配置参数说明表**中的

“enableLocalTest”参数，当该参数为true时，上报日志规格为单个机器200次/秒（即每秒只能发送200次，每次批量发送数量/大小详情见参数“batchSizeThresholdInBytes、batchCountThreshold、lingerMs”）。

- 支持将logback.xml变为logback-spring.xml（SpringBoot），且读取springProfile中的配置。
- 支持读取logback.xml、logback-spring.xml中的自定义标签。
- 通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

注意事项

说明

建议设置上报周期小于1分钟（例如50秒）上报一次，设置1分钟或超过1分钟上报一次，可能会出现连接失败，因为SDK连接默认1分钟超时。

由于LogBack-SDK默认对SK明文存储，不符合某些对于安全有更高要求的用户，lts提供了一种用户自定义的加解密方式，建议用户优先使用加解密方式。

加密过程如下：

1. 用户编写一个java类，比如com.demo.DecryptDemo，需要继承logback.core包中的PropertyDefinerBase类，重写getPropertyValue方法。
2. 用户在getPropertyValue方法中实现SK的解密，返回解密后的值。
3. 当用户调用LogBack-SDK初始化时，需要传入SK，这时在logback.xml中定义变量如下：
```
<define name="customerSk" class="com.huawei.log.DecryptSkDemo"/>
```
4. 传入SK时，定义变量值如下：
```
<accessKeySecret>${customerSk}</accessKeySecret>
```

安装 SDK

您可以通过以下两种方式安装日志服务LogBack SDK

方式一：在Maven项目中加入依赖项（推荐方式）。

1. maven构建时，settings.xml文件的mirrors节点需要增加**华为外部开源仓**。如下：
```
<mirror>
  <id>huaweicloud</id>
  <mirrorOf>*</mirrorOf>
  <url>https://repo.huaweicloud.com/repository/maven/</url>
</mirror>
```
2. 您可以在**华为外部开源仓**中获取日志服务LogBack SDK依赖的最新版本。
3. 在Maven工程中使用日志服务LogBack SDK，只需在pom.xml中加入相应依赖即可，Maven项目管理工具会自动下载相关JAR包。以1.0.1版本为例，在<dependencies>中加入如下内容：
```
<dependency>
   <groupId>io.github.huaweicloud</groupId>
   <artifactId>lts-sdk-common</artifactId>
   <version>1.0.1</version>
<exclusions>
      <exclusion>
         <groupId>org.slf4j</groupId>
         <artifactId>slf4j-simple</artifactId>
      </exclusion>
```

```
    </exclusions>
</dependency>
<dependency>
    <groupId>io.github.huaweicloud</groupId>
    <artifactId>lts-sdk-logback</artifactId>
    <version>1.0.1</version>
</dependency>
```

方式二：在项目中直接依赖LogBack SDK的jar包。

1. 下载**lts-sdk-common**和**lts-sdk-logback**包。
2. 请确保您的项目中有以下依赖，因为SDK的jar包中需要该依赖。

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.83</version>
</dependency>
<dependency>
    <groupId>joda-time</groupId>
    <artifactId>joda-time</artifactId>
    <version>2.10.14</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.11</version>
</dependency>
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.5.13</version>
</dependency>
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>30.1.1-jre</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-classic</artifactId>
    <version>1.2.11</version>
</dependency>
<dependency>
    <groupId>ch.qos.logback</groupId>
    <artifactId>logback-access</artifactId>
    <version>1.2.11</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.11.0</version>
</dependency>
```

示例代码

1. 编写logback.xml、logback-spring.xml中LogbackSDKAppender，示例如下：

```
<configuration scan="true" debug="false">
  <appender name="LogInfoTest"
class="com.huaweicloud.lts.appender.LogbackSDKAppender">
    <!--  华为云账号的项目ID（ project id ）  -->
    <projectId>xxxxxxxxxxxxxxxxxxxxxx</projectId>
```

```
    <!--  华为云日志服务日志组ID  -->
    <logGroupId>xxxxxxxxxxxxxxxxxxxxxx</logGroupId>
    <!--  华为云日志服务日志流ID  -->
    <logStreamId>xxxxxxxxxxxxxxxxxxxxxx</logStreamId>
    <!--  华为云账号的AK, 认证用的AK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放, 使用时解密, 确保安全  -->
    <accessKeyId>xxxxxxxxxxxxxxxxxxxxxx</accessKeyId>
    <!--  华为云账号的SK, 认证用的SK硬编码到代码中或者明文存储都有很大的安全风险, 建议密文存放, 使用时解密, 确保安全  -->
    <accessKeySecret>xxxxxxxxxxxxxxxxxxxxxx</accessKeySecret>
    <!--  云日志服务的区域  -->
    <regionName>cn-nxxxxx</regionName>
    <encoder>
      <pattern>%d %-5level [%thread] [%customerParam] %logger{0}: %msg</pattern>
    </encoder>
    <!--  单个Appender能缓存的日志大小上限  -->
    <totalSizeInBytes>104857600</totalSizeInBytes>
    <!--  producer发送日志时阻塞时间  -->
    <maxBlockMs>0</maxBlockMs>
    <!--  执行日志发送任务的线程池大小  -->
    <ioThreadCount>8</ioThreadCount>
    <!--  producer发送单批日志量上限  -->
    <batchSizeThresholdInBytes>524288</batchSizeThresholdInBytes>
    <!--  producer发送单批日志条数上限  -->
    <batchCountThreshold>4096</batchCountThreshold>
    <!--  producer发送单批日志等待时间  -->
    <lingerMs>2000</lingerMs>
    <!--  producer发送日志失败后重试次数  -->
    <retries>5</retries>
    <!--  首次重试的退避时间  -->
    <baseRetryBackoffMs>500</baseRetryBackoffMs>
    <!--  重试的最大退避时间  -->
    <maxRetryBackoffMs>500</maxRetryBackoffMs>
    <!-- 默认false, true: 可以跨云上报日志, false: 仅能在华为云ecs主机上报日志 -->
    <enableLocalTest>false</enableLocalTest>
    <!-- 超过1M的日志, 拆分后放弃1M以外的 -->
    <!-- <giveUpExtraLongSingleLog>true</giveUpExtraLongSingleLog>-->
  </appender>
  <logger name="com.huawei.test.logback.info" level="INFO">
    <appender-ref ref="LogInfoTest"/>
  </logger>
</configuration>
```

2. 编写代码中产生日志，示例如下：

```
package com.huawei.test.logback.info;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class TestLogBackAppenderInfo {

    private static final Logger LOGGER =
LoggerFactory.getLogger(TestLogBackAppenderInfo.class);

    public void run() throws InterruptedException {
        LOGGER.info("test~logback~~~info");
    }

}
```


配置参数说明

- Appender配置参数说明

表 4-132 Appender 配置参数说明表

参数名称	描述	类型	是否需要填写	默认值
projectId	华为云账号的项目ID（project id）。	String	必填	-
accessKeyId	华为云账号的AK。认证用的AK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
accessKeySecret	华为云账号的SK。认证用的SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。	String	必填	-
regionName	云日志服务的区域。	String	必填	-
logGroupId	LTS的日志组ID。	String	必填	-
logStreamId	LTS的日志流ID。	String	必填	-
appName	日志标签appName字段。	String	选填	-
pathFile	日志标签pathFile字段。	String	选填	-
totalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	100M（100 * 1024 * 1024）

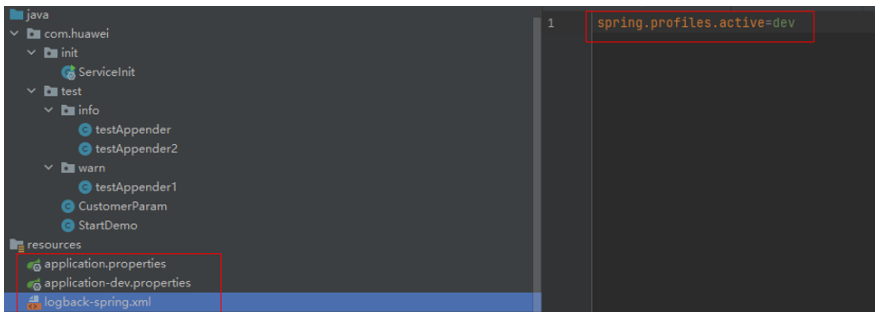
参数名称	描述	类型	是否需要填写	默认值
maxBlockMs	如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，单位为毫秒。默认为 60 秒（60000毫秒），建议为0秒。 ● 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 ● 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60秒（60000毫秒）
ioThreadCount	执行日志发送任务的线程池大小。	int	选填	客户机器空闲CPU数量，但一定大于1
batchSizeThresholdInBytes	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	0.5M（512 * 1024）
batchCountThreshold	当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送。	int	选填	4096
lingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2S
retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 5 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10
baseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1S
maxRetryBackoffMs	重试的最大退避时间。	long	选填	50S
giveUpExtraLongSingleLog	超过1M的日志, 拆分后丢弃大于1M的数据。	boolean	选填	false

参数名称	描述	类型	是否需要填写	默认值
enableLocalTest	是否开启跨云上报日志。 • 选true时，开启跨云后用户能通过公网（仅支持华北-北京四、华东-上海一、华南-广州、西南-贵阳一）访问（建议调测使用）。 • 选false时，关闭跨云后用户是通过华为云当前区域主机访问。	boolean	选填	false

- 如果是SpringBoot项目，则可以将logback.xml替换为logback-spring.xml，且支持读取springProfile中的配置，可以自由切换开发、测试等环境的日志采集环境。

  使用说明如下：

  a. 在SpringBoot项目中的application.properties可以自由指定使用的配置项。

  b. logback-spring.xml中配置

```
<springProfile name="dev">
    <root level="INFO">
        <appender-ref ref="customerAppender"/>
    </root>
</springProfile>
```

- 如果需要logback.xml、logback-spring.xml支持自定义标签，且将标签中的value打印到日志中。

  使用说明：

  a. 需要继承ClassicConverter类，重写convert方法，返回值为自定义标签的value。

```
public class CustomerParam extends ClassicConverter {
    @Override
    public String convert(ILoggingEvent iLoggingEvent) {
        return "自定义标签value";
    }
}
```

  b. 在logback.xml或者logback-spring.xml中自定义a中的标签类，使用%自定标签的名称，来使用此标签。

```
<conversionRule conversionWord="customerParam"
converterClass="com.huawei.test.logback.CustomerParam"/>

    <appender name="customer" class="com.huawei.appender.LogbackSDKAppender">
        <encoder>
```



```
        <pattern>%d %-5level [%thread] [%customerParam] %logger{0}: %msg</pattern>
    </encoder>
</appender>
```

参数获取方法

- 区域表

  表 4-133 区域表

区域名称	RegionName
华北-北京二	cn-north-2
华北-北京四	cn-north-4
华北-北京一	cn-north-1
华东-上海二	cn-east-2
华东-上海一	cn-east-3
华南-广州	cn-south-1
华南-深圳	cn-south-2
西南-贵阳一	cn-southwest-2
亚太-新加坡	ap-southeast-3

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.4 云日志服务 Go SDK

云日志服务SDK提供了Go语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为云账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。

使用说明

当用户修改权限后，权限信息在一天后生效。通过SDK上报日志到LTS的时间相距当前时间不超过2天，否则上报日志会被LTS删除。

注意事项

认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全。

安装 Go SDK

步骤1 获取LTS Go SDK包。

```
go get github.com/huaweicloud/huaweicloud-lts-sdk-go
```

步骤2 引用日志LTS Go SDK包。

```
import github.com/huaweicloud/huaweicloud-lts-sdk-go
```

步骤3 编写上报日志代码，示例如下：

说明

可以自定义发送成功与失败时的回调方法，实现方式请参考如下命令：自定义ErrorHandler的结构体，实现Success和Fail方法，并且在发送日志时直接使用。

```
SendLogWithCallBack(groupId, streamId string, log *Log, callback CallBack)
```

```
package main

import (
"github.com/huaweicloud/huaweicloud-lts-sdk-go"
"fmt"
"sync"
"time"
)

var (
ak        = ""
sk        = ""
pid       = ""
groupId   = ""
streamId1 = ""
streamId2 = ""
region    = ""
endpoint  = ""
)

func main() {
// 获取默认配置
producerConfig := producer.GetConfig()
// 上报地址
producerConfig.Endpoint = endpoint
// accessKeyId：华为云账号的AK, 认证用的AK硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密文存放, 使用时解密, 确保安全
producerConfig.AccessKeyID = os.Getenv("accessKeyId")
// accessKeySecret：华为云账号的SK, 认证用的SK硬编码到代码中或者明文存储都有很大的安全风险, 建议在配置文件或者环境变量中密文存放, 使用时解密, 确保安全
producerConfig.AccessKeySecret = os.Getenv("accessKeySecret")
// 区域名称
producerConfig.RegionId = region
// 账户projectid
producerConfig.ProjectId = pid
// 初始化producer实例
producerInstance := producer.InitProducer(producerConfig)
// 启动producer
producerInstance.Start()
```

```
wg := sync.WaitGroup{}
for i := 0; i < 10; i++ {
wg.Add(1)
go func() {
for j := 0; j < 1000; j++ {
labels := make(map[string]string)
labels["keyA"] = "valueA"
labels["keyB"] = "valueB"
labels["keyC"] = "valueC"

logContent := fmt.Sprintf("content for this test %d", j)
// 生成日志 填入日志内容及labels
log := producer.GenerateLog([]string{logContent}, labels)
// 发送日志到日志组，日志流
err := producerInstance.SendLog(groupId, streamId1, log)
err = producerInstance.SendLog(groupId, streamId2, log)
if err != nil {
fmt.Println(err)
}

handle := ErrorHandle{}
// 带CallBack的发送日志方法，发送失败时会调用ErrorHandler的Fail方法
err = producerInstance.SendLogWithCallBack("groupId", "streamId1", log, handle)
err = producerInstance.SendLogWithCallBack("groupId", "streamId1", log, handle)
if err != nil {
    fmt.Println(err)
}

time.Sleep(1 * time.Microsecond)
}

wg.Done()
fmt.Printf("test func finished\n")
}()
}

wg.Wait()
fmt.Printf("send all complate ...")
// 关闭发送实例
producerInstance.Close(60 * 1000)
time.Sleep(10 * 60 * time.Second)
}

// errorHandle，可以针对发送成功和失败进行相关的callback调用，方便失败时打印错误日志，进行问题定位
type ErrorHandle struct{}

// 发送成功时的callback方法
func (ErrorHandle) Success(result *producer.Result) {
    fmt.Printf("send log to lts success, success flag: %v\n", result.IsSuccessful())
}

// 发送失败时的callback方法，可以打印result中的错误信息，方便问题定位
func (ErrorHandle) Fail(result *producer.Result) {
    fmt.Printf("send log to lts error, success flag: %v requestId: %s, httpcode: %d, errorCode: %s, errorMsg:
%s\n",
        result.IsSuccessful(), result.GetRequestId(), result.GetHttpCode(), result.GetErrorCode(),
result.GetErrorMessage())
}
```

----结束

配置参数说明

- producer config参数说明

参数名称	描述	类型	是否需要填写	默认值
ProjectId	华为云账号的项目ID（project id）。	String	必填	-
AccessKeyId	华为云账号的AK。	String	必填	-
AccessKeySecret	华为云账号的SK。	String	必填	-
RegionName	云日志服务的区域。	String	必填	-
Endpoint	上报目的。	String	必填	-
TotalSizeInBytes	单个producer实例能缓存的日志大小上限。	int	选填	100M（100 * 1024 * 1024）
MaxBlockSec	如果 producer 可用空间不足，调用者在 send 方法上的最大阻塞时间，默认为 60 秒。建议为0秒。 当maxBlockMs值>=0时，则阻塞到设置的时间，如果到达阻塞时间，还是不能获取到内存，即报错且丢弃日志。 当maxBlockMs值=-1时，则一直阻塞到发送成功，且不会丢弃日志。	long	选填	60S
MaxIoWorkers	执行日志发送任务的任务池大小。	int	选填	默认10个
MaxBatchSize	当一个 ProducerBatch 中缓存的日志大小大于等于 batchSizeThresholdInBytes 时，该 batch 将被发送。	int	选填	0.5M（512 * 1024）
MaxBatchCount	当一个 ProducerBatch 中缓存的日志条数大于等于 batchCountThreshold 时，该 batch 将被发送。	int	选填	4096
LingerMs	一个 ProducerBatch 从创建到可发送的逗留时间。	int	选填	2S

参数名称	描述	类型	是否需要填写	默认值
Retries	如果某个 ProducerBatch 首次发送失败，能够对其重试的次数，建议为 3 次。如果 retries 小于等于 0，该 ProducerBatch 首次发送失败后将直接进入失败队列。	int	选填	10
BaseRetryBackoffMs	首次重试的退避时间。	long	选填	0.1S
MaxRetryBackoffMs	重试的最大退避时间。	long	选填	50S

- 日志生成GenerateLog方法类参数说明

参数名称	描述	类型	是否需要填写
contents	批量日志内容。	[]string	是
labels	日志标签，map格式。	map[string]string	是

说明

单次上报条数小于4096条，且小于512K。

- 日志上报SendLog方法类参数说明

参数名称	描述	类型	是否需要填写
groupId	日志组id	long	是
streamId	日志流id	String	是
log	日志结构	struct log	是

参数获取方式

- 上线区域表，使用华为云主机时参考以下信息：

区域名称	RegionName	Endpoint
华北-北京四	cn-north-4	https://lts-access.cn-north-4.myhuaweicloud.com:8102
华东-上海一	cn-east-3	https://lts-access.cn-east-3.myhuaweicloud.com:8102

区域名称	RegionName	Endpoint
华南-广州	cn-south-1	https://lts-access.cn-south-1.myhuaweicloud.com:8102
亚太-新加坡	ap-southeast-3	https://lts-access.ap-southeast-3.myhuaweicloud.com:8102

说明

当前仅华北-北京四、华东-上海一、华南-广州区域使用SDK时支持跨云/本地上报日志，Endpoint端口使用443。

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.5 云日志服务 Web SDK

云日志服务Web SDK提供了Web小程序上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务Web SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**日志流**。

版本更新说明

表 4-134 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。

版本号	更新说明
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例。

使用说明

- Web SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。
- 为确保Web小程序日志上报正常，需将上报地址添加到Web开发者平台域名列表，上报地址请参见**配置参数说明**。

NPM 方式安装 SDK

1. 安装方法：在项目根目录下通过运行“npm i lts-web-sdk”命令，安装SDK软件包。

   说明

   您可以在**开源仓地址**下载最新的SDK。

2. 示例代码：

```
const LTS_WEB_SDK = require('lts-web-sdk').default;
// import LTS_WEB_SDK from 'lts-web-sdk';
// 初始化
const weblog = new LTS_WEB_SDK({
  // 上报region
  region: string,
  // 华为云项目ID
  projectId: string,
  // 上报地址
  url: string,
  // LTS日志组ID
  groupId: string,
  // LTS日志流ID
  streamId: string,
  // 调试日志等级
  debug: string,
  // 上报条数阈值
  cacheThreshold: number,
  // 上报时间阈值
  timeInterval: number,
});
// 立即上报单条带标签
weblog.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
weblog.reportImmediately([{  key: 'value',  number: 1,  array: [],  json: {  json: 'json'  } }, { 'key': 'value' }]);
// 缓存上报多条 带标签
weblog.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
// 缓存上报多条 不带标签
weblog.report([{ 'name': 'xiaohong', 'age': 18 }, {  key: 'value',  number: 1,  array: [],  json: {  json: 'json'  } }]);
// 缓存上报多条 带多个标签（最多50个）
weblog.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'web', 'link': '/', from: 'web'});
```

CDN 同步方式安装 SDK

1. 安装方法：

   在您的html文件中，通过以下方式引用SDK。

   <script src="https://res.hc-cdn.com/web-sdk-cdn/版本号/websdk.min.js"></script>

2. 示例代码：

```
// 页面引入SDK
<script src="https://res.hc-cdn.com/web-sdk-cdn/1.0.15/websdk.min.js"></script>
// 初始化
const weblog = new LTS_WEB_SDK({
  // 上报region
  region: string,
  // 华为云项目ID
  projectId: string,
  // 上报地址
  url: string,
  // LTS日志组ID
  groupId: string,
  // LTS日志流ID
  streamId: string,
  // 调试开关，开启后可以看到输出的调试日志
  debug: boolean,
  // 上报条数阈值
  cacheThreshold: number,
  // 上报时间条数
  timeInterval: number,
});
// 立即上报单条带标签
weblog.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
// 立即上报单条 不带标签
weblog.reportImmediately([{   key: 'value',   number: 1,   array: [],   json: {     json: 'json'   } }, { 'key': 'value' }]);
// 缓存上报多条 带标签
weblog.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
// 缓存上报多条 不带标签
weblog.report([{ 'name': 'xiaohong', 'age': 18 }, {   key: 'value',   number: 1,   array: [],   json: {     json: 'json'   } }]);
// 缓存上报多条 带多个标签（最多50个）
weblog.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'web', 'link': '/', from: 'web'});
```

配置参数说明

- 配置参数说明

表 4-135 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。

url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3

区域名称	RegionName
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.6 云日志服务 iOS SDK

云日志服务iOS SDK提供了Objective-C & Swift语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

说明

当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为账号，并开通云日志服务。

  说明

  当用户修改权限后，权限信息在一天后生效。
- 确认云日志服务的区域，请用户根据所在区域，选择region。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。

版本更新说明

说明

SDK如何处理个人信息请参考**华为云日志服务移动端日志采集SDK隐私声明**。

您集成和使用我们的SDK时需要遵从个人信息保护基本要求，详情请参考**华为云日志服务移动端日志采集SDK开发者合规指南**。

表 4-136 版本更新说明

版本号	下载地址	检验信息下载地址	更新说明	系统
1.0.27	单击**下载**	单击**下载**	优化性能。	iOS10、Xcode11及以上

版本号	下载地址	检验信息下载地址	更新说明	系统
1.0.26	单击下载	单击下载	优化性能。	iOS10、Xcode11及以上
1.0.24	单击下载	单击下载	1. 支持更多region：华东-上海一、华南-广州。 2. 优化内存使用、日志发送等。	iOS10、Xcode11及以上
1.0.21	单击下载	单击下载	1. 优化初始化函数initWithConfig。 2. 为配置函数config添加废弃声明。	iOS10、Xcode11及以上
1.0.19	单击下载	单击下载	1. 调整配置项阈值。 2. 修复兼容性问题。	iOS10、Xcode11及以上
1.0.18	单击下载	单击下载	1. 增加设置调试日志级别接口：setLogLevel。 2. 在控制台输出配置项、上报接口参数、请求发送内容。 3. 上报接口参数labels支持嵌套。 4. 请求发送控制、数据库大小限制等优化。	iOS10、Xcode11及以上
1.0.15	单击下载	单击下载	首次发布，提供四个接口：初始化initWithConfig、配置 config、上报日志 report、立即上报日志reportImmediately。	iOS10、Xcode11及以上。

安装 iOS SDK

步骤1 集成接入SDK。

方法1：通过CocoaPods集成

- 在Podfile中添加CocoaPods官方仓库
  ```
  source 'https://github.com/CocoaPods/Specs.git'
  ```
- 在Podfile中添加依赖
  ```
  pod 'LTSSDK' , '1.0.27'
  ```
- 在终端执行
  ```
  pod install --repo-update
  ```

方法2：手动集成

- 下载SDK，解压到指定目录。注：直接解压即可，不需要额外操作。
- 将解压后的xcframework静态库添加到您的项目工程中。

说明

待导入的xcframework静态库必须和工作空间在相同的磁盘空间里，如果不在，您可以选择注意勾选“Copy items if needed”和“Create groups”，将待导入的xcframework库工程文件复制到工作空间。

步骤2 初始化，详细参数请参考表4-139。

- Objective-C：初始化代码示例。

```
// LTS参数配置
LTSConfigParams *params = [[LTSConfigParams alloc] init];
// 必填参数
params.region = @"云日志服务的区域";
params.projectId = @"华为云账号的项目ID";
params.groupId = @"LTS的日志组ID";
params.streamId = @"LTS的日志流ID";
// 注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
params.accessKey = @"华为云访问密钥";
params.secretKey = @"华为云秘密访问密钥";
// 选填参数
// params.url = @"https://lts-access.cn-north-4.myhuaweicloud.com"; //要上报的LTS公网地址域名
// params.cacheThreshold = 200;
// params.timeInterval = 3;
// params.reportWhenEnterBackgroundEnabled = YES;
// params.reportWhenAPPLaunchEnabled = NO;
// LTS初始化方法
LTSSDK *lts = [[LTSSDK alloc] initWithConfig:params];
```

- Swift：初始化代码示例。

```
// LTS参数配置
let params = LTSConfigParams()
// 必填参数
params.region = "云日志服务的区域"
params.projectId = "华为云账号的项目ID"
params.groupId = "LTS的日志组ID"
params.streamId = "LTS的日志流ID"
// 注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
params.accessKey = "华为云访问密钥"
params.secretKey = "华为云秘密访问密钥"
// 选填参数
// params.url = "https://lts-access.cn-north-4.myhuaweicloud.com" //要上报的LTS公网地址域名
// params.cacheThreshold = 200
// params.timeInterval = 3
// params.reportWhenEnterBackgroundEnabled = true
// params.reportWhenAPPLaunchEnabled = false
// LTS初始化方法
let lts = LTSSDK(config:params)
```

说明

LTSSDK支持多实例日志上报机制，根据不同配置创建实例进行上报。

步骤3 上报日志。

LTS提供两种上报日志的方法。

表 4-137 上报日志的方法

Method	Description
- (BOOL)report:(id)content labels:(nullable NSDictionary<NSString *, id> *)labels;	上报日志：先存入本地数据库，根据配置中设定的策略实施上报。

Method	Description
- (BOOL)reportImmediately:(id)content labels:(nullable NSDictionary<NSString *, id> *)labels;	立即上报日志。

表 4-138 Parameters 参数

Name	Description
content	日志内容。支持字典或字典数组；最外层键值对最多300个；字典转JSON字符串最大支持长度为30720，超出部分将被截断再上报。
labels	日志标签。支持字典或空值；最外层键值对最多50个；最外层key最大长度为64，支持字母、数字和下划线组合，首字符须是字母；字典转JSON字符串最大支持长度为30720，超出则该条日志无法上报。

Objective-C代码示例如下：

```
//单个上报
[lts report:@{@"content_key": @"content_value"} labels:@{@"label_key": @"label_value"}];
[lts reportImmediately:@{@"content_key": @"content_value"} labels:@{@"label_key": @"label_value"}];

//批量上报
[lts report:@[@{@"content1_key": @"content1_value"}, @{@"content2_key": @"content2_value"}]
labels:@{@"label_key": @"label_value"}];
[lts reportImmediately:@[@{@"content1_key": @"content1_value"}, @{@"content2_key":
@"content2_value"}] labels:@{@"label_key": @"label_value"}];
```

Swift代码示例如下：

```
//单个上报
lts.report(["content_key": "content_value"], labels: ["label_key": "label_value"])
lts.reportImmediately(["content_key": "content_value"], labels: ["label_key": "label_value"])

//批量上报
lts.report([["content1_key": "content1_value"], ["content2_key": "content2_value"]], labels: ["label_key":
"label_value"])
lts.reportImmediately([["content1_key": "content1_value"], ["content2_key": "content2_value"]], labels:
["label_key": "label_value"])
```

----结束

配置参数说明

表 4-139 初始化参数说明

参数名称	类型	是否必填	默认值	描述
projectId	NSString	必填	-	华为云账号的项目ID。

参数名称	类型	是否必填	默认值	描述
accessKey	NSString	必填	-	华为云账号的访问密钥，简称AK。注意：硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
secretKey	NSString	必填	-	华为云账号的秘密访问密钥，简称SK。注意：硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
region	NSString	必填	-	LTS的服务区域。
groupId	NSString	必填	-	LTS的日志组ID。
streamId	NSString	必填	-	LTS的日志流ID。
url	NSString	选填	nil	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com
cacheThreshold	NSUInteger	选填	200条	当本地数据库日志存储条数达到该值会触发上报，取值范围为30-1000条。
timeInterval	NSUInteger	选填	3秒	定时器每隔该值会触发上报，取值范围为1-1800秒。
reportWhenEnterBackgroundEnabled	BOOL	选填	YES	是否开启APP切后台上报功能。
reportWhenAPPLaunchEnabled	BOOL	选填	NO	是否开启APP启动时上报功能。如需开启，请在UIApplicationDidFinishLaunchingNotification通知发送前完成配置。

上报日志策略说明

LTSSDK支持四种日志上报策略：阈值上报、定时上报、应用切后台和启动时上报，根据场景选择合适的值配置cacheThreshold、timeInterval、reportWhenEnterBackgroundEnabled、reportWhenAPPLaunchEnabled等参数。

开启调试

在开发过程中可以启用调试模式，借助控制台日志实时查看上报日志的记录情况，观察具体结果并根据需要进行调整。

在初始化LTSSDK之前，调用类方法+(void)setLogLevel:(LTSLoggerLevel)logLevel设置调试级别：Debug、Info、Warn、Error、Off。

Objective-C语言示例代码：

```
#ifdef DEBUG
[LTSSDK setLogLevel:LTSLoggerLevelDebug];
#endif
```

Swift语言示例代码：

```
#if DEBUG
LTSSDK.setLogLevel(.debug)
#endif
```

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.7 云日志服务 Android SDK

云日志服务Android SDK提供了Kotlin & Java语言上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

说明

当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。

传输协议

HTTPS

使用前提

- 使用云日志SDK前，您需要注册华为账号，并开通云日志服务。

  说明

  当用户修改权限后，权限信息在一天后生效。
- 确认云日志服务的区域，请用户根据所在区域，选择region。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。

版本更新说明

说明

SDK如何处理个人信息请参考**华为云日志服务移动端日志采集SDK隐私声明**。

您集成和使用我们的SDK时需要遵从个人信息保护基本要求，详情请参考**华为云日志服务移动端日志采集SDK开发者合规指南**。

表 4-140 版本更新说明

版本号	下载地址	检验信息下载地址	更新说明	系统
1.0.27	单击**下载**	单击**下载**	1. 优化了缓存效率	Android 7 及以上
0.0.4	请通过maven Central下载。	-	1. 功能与1.0.27版本相同，可支持kotlin 1.3.x	Android 7 及以上
1.0.26	单击**下载**	单击**下载**	1. 修复了因异步修改入参导致数据异常的问题	Android 7 及以上
1.0.25	单击**下载**	单击**下载**	1. 增强代码强壮性，添加参数保护	Android 7 及以上
1.0.24	单击**下载**	单击**下载**	1. 支持更多region：华东-上海一、华南-广州 2. 修改了程序目录，修改为com.cloud.lts.*	Android 7 及以上
1.0.21	单击**下载**	单击**下载**	1. 废弃setconfig方法，使用LTSSDK的构造方法代替。	Android 7 及以上
1.0.19	单击**下载**	单击**下载**	1. 修改了上报时间间隔阈值。 2. 修复了修改配置信息后导致无法获取缓存的问题。	Android 7 及以上

版本号	下载地址	检验信息下载地址	更新说明	系统
1.0.18	单击**下载**	单击**下载**	1. 增强代码强壮性，修复多实例时可能会产生的崩溃。 2. 修复最低 Android API 支持版本定义错误的问题。 3. 增加设置调试日志级别接口：setLogLevel。 4. 修改了 日志发送的入参 label 的类型，支持多层嵌套。 5. 日志发送入参 content 支持数组类型，方便使用。 6. 添加缓存存储条目最大值支持，如果超过会丢弃后再存储。 7. 修改了请求失败时的控制策略，支持递进式等待。 8. 对本地错误日志输出做了整理。	Android 7 及以上
1.0.17	单击**下载**	单击**下载**	1. 首次发布，提供四个接口：初始化、配置config、上报日志report、立即上报日志reportImmediately。	Android 7 及以上

安装 Android SDK

步骤1 集成接入SDK。

1. maven仓库集成。
   - 在build.gradle中添加依赖。

```
plugins {
    ...
    id 'org.jetbrains.kotlin.android' version 'KOTLIN_VERSION' apply false  // 添加kotlin插件，1.x.x最低支持1.6.20版本，最低可支持1.3.61版本
    ...
}
```

   - 在app/build.gradle中添加依赖。

```
dependencies {
    ...
    implementation 'io.github.lts-sdk:lts-sdk-android:1.0.27'
    ...
}
```

   - 在 settings.gradle中添加maven仓库源。

```
pluginManagement{
    ...
    repositories {
        ...
        // 加入下面内容
        mavenCentral()
    }
}
dependencyResolutionManagement {
```



```
    ...
    repositories {
        ...
        // 加入下面内容
        mavenCentral()
    }
}
```

2. 手动集成。
   - 下载日志SDK包。
   - 下载后解压到指定目录。注：直接解压即可，不需要额外操作。
   - 将解压后的aar静态库文件添加到您的项目工程中。
   - 在build.gradle中添加依赖。

```
plugins {
    ...
    id 'org.jetbrains.kotlin.android' version 'KOTLIN_VERSION' apply false  // 添加kotlin插件，1.x.x最低支持1.6.20版本，0.0.x最低可支持1.3.61版本
    ...
}
```

   - 在app/build.gradle中添加依赖。

```
dependencies {
    ...
    implementation fileTree(dir: $dir, include: ['*.aar']) // 填写aar所在的文件夹，例如'libs'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.4.0' // 0.0.x版本依赖版本为1.3.3
    implementation 'androidx.core:core-ktx:1.7.0'// 0.0.x版本依赖版本为1.2.0
    implementation 'com.google.code.gson:gson:2.8.9'
    kapt 'androidx.room:room-compiler:2.3.0'     // 0.0.x版本依赖版本为2.2.6
    api 'androidx.room:room-runtime:2.3.0'       // 0.0.x版本依赖版本为2.2.6
    api 'androidx.room:room-common:2.3.0'        // 0.0.x版本依赖版本为2.2.6
    ...
}
```

   - 在 settings.gradle中添加maven仓库源。

```
pluginManagement{
    ...
    repositories {
        ...
        // 加入下面内容
        google()
        mavenCentral()
    }
}
dependencyResolutionManagement {
    ...
    repositories {
        ...
        // 加入下面内容
        google()
        mavenCentral()
    }
}
```

说明

- 如果你的Gradle版本低于7.0 则需要将mavan仓库源添加到build.gradle文件中。

步骤2 初始化，详细参数请参考表4-143。

- 添加依赖

```
import com.cloud.lts.UserConfig
import com.cloud.lts.LTSSDK
```

- Kotlin：初始化代码示例。

```
// 注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
val ak: String = getData("HUAWEICLOUD_SDK_AK")
```



```
val sk: String = getData("HUAWEICLOUD_SDK_SK")
// LTS参数配置
val userConfig = UserConfig.Builder()
// 必填参数
.setRegion(USER_REGION)  // 云日志服务的区域
.setProjectId(PROJECT_ID)  // 华为云账号的项目ID
.setGroupId(LOG_GROUP_ID)  // LTS的日志组ID
.setStreamId(LOG_STREAM_ID)  // LTS的日志流ID
.setAccessKey(ak)  // 华为云访问密钥
.setSecretKey(sk)  // 华为云秘密访问密钥
// 选填参数
.setUrlHost(url)  // LTS的公网地址域名
.setCacheThreshold(200)
.setTimeInterval(60)
.setIsReportBackground(true)
.setIsReportLaunch(false)
.build()
 // LTS初始化方法
ltssdk = LTSSDK( </* Application Instance */> , userConfig)
```

- Java：初始化代码示例。

```
// 注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
String ak = getData("HUAWEICLOUD_SDK_AK")
String sk = getData("HUAWEICLOUD_SDK_SK")
// LTS参数配置
UserConfig userConfig = new UserConfig.Builder()
 // 必填参数
.setRegion(USER_REGION)  // 云日志服务的区域
.setProjectId(PROJECT_ID)  // 华为云账号的项目ID
.setGroupId(LOG_GROUP_ID)  // LTS的日志组ID
.setStreamId(LOG_STREAM_ID)  // LTS的日志流ID
.setAccessKey(ak)  // 华为云访问密钥
.setSecretKey(sk)  // 华为云秘密访问密钥
// 选填参数
.setUrlHost(url)  // LTS的公网地址域名
.setCacheThreshold(200)
.setTimeInterval(60)
.setIsReportBackground(true)
.setIsReportLaunch(false)
.build();
// LTS初始化方法
ltssdk = new LTSSDK( </* Application Instance */> , userConfig);
```

说明

LTSSDK支持多实例日志上报机制，根据不同配置创建实例进行上报。

步骤3 上报日志。

LTS提供两种上报日志的方法。

表 4-141 上报日志的方法

Method	Description
report(content, labels)	上报日志：先存入本地数据库，根据配置中设定的策略实施上报。
reportImmediately(content, labels)	立即上报日志。

表 4-142 Parameters 参数

Name	Description
content	日志内容，支持字典和字典数组；键值对最多300个；content转JSON字符串最大支持长度为30*1024，超出部分被截断。
labels	日志标签。支持字典或空值；最外层键值对最多50个；最外层key最大长度为64，支持字母、数字和下划线组合，首字符须是字母；字典转JSON字符串最大支持长度为30720，超出则该条日志无法上报。

Kotlin代码示例如下：

```
var ltssdk: LTSSDK
val food = hashMapOf(Pair("food_1", "rice"), Pair("food_2", "wheat"), Pair("food_3", "egg"))
val fruit = hashMapOf(Pair("fruit_1", "apple"), Pair("fruit_2", "pear"), Pair("fruit_3", "banana"))
val label = hashMapOf( Pair("date","2023-10-01"))
val contents = listOf(food, fruit).toTypedArray()
ltssdk.report(food, label)    // 缓存上报单条 带标签
ltssdk.report<String, String>(food)    // 缓存上报单条 不带标签
ltssdk.reportImmediately(food, label)    // 立即上报单条 带标签
ltssdk.reportImmediately<String, String>(food)    // 立即上报单条 不带标签
ltssdk.report(contents, label)    // 缓存上报多条 带标签
ltssdk.report<String, String>(contents)    // 缓存上报多条 不带标签
ltssdk.reportImmediately(contents , label)    // 立即上报多条 带标签
ltssdk.reportImmediately<String, String>(contents)    // 立即上报多条 不带标签
```

Java代码示例如下：

```
LTSSDK ltssdk;
HashMap fruit = new HashMap<String, String>(){{
    put("fruit_1", "apple");
    put("fruit_2", "pear");
    put("fruit_3", "banana");
}};
HashMap food = new HashMap<String, String>(){{
    put("food_1", "rice");
    put("food_2", "wheat");
    put("food_3", "egg");
}};
HashMap<String, String> labels = new HashMap(){{
    put("date","2023-10-01");
}};
HashMap[] contents = {food, fruit};
ltssdk.report(food);  // 缓存上报单条 不带标签
ltssdk.report(food, labels);  // 缓存上报单条 带标签
ltssdk.reportImmediately(food);  // 立即上报单条 不带标签
ltssdk.reportImmediately(food, labels);  // 立即上报单条 带标签
ltssdk.report(contents); // 缓存上报多条 不带标签
ltssdk.report(contents, labels); // 缓存上报多条 带标签
ltssdk.reportImmediately(contents);  // 立即上报多条 不带标签
ltssdk.reportImmediately(contents, labels);  // 立即上报多条 带标签
```

----结束

配置参数说明

表 4-143 初始化参数说明

参数名称	类型	是否必填	默认值	描述
projectId	String	必填	-	华为云账号的项目ID。
accessKey	String	必填	-	华为云账号的访问密钥，简称AK。注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
secretKey	String	必填	-	华为云账号的秘密访问密钥，简称SK。注意：认证用的华为云账号AK、SK硬编码到代码中或者明文存储都有很大的安全风险，建议密文存放，使用时解密，确保安全。
region	String	必填	-	LTS的服务区域。
groupId	String	必填	-	LTS的日志组ID。
streamId	String	必填	-	LTS的日志流ID。
url	String	选填	null	用于上报的公网地址域名，如未设置，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com
cacheThreshold	Long	选填	200条	当本地数据库日志存储条数达到该值会触发上报，取值范围为30-1000条。
timeInterval	Long	选填	3秒	定时器每隔该值会触发上报，取值范围为1-1800秒。
isReportBackground	boolean	选填	true	是否开启APP切入后台时上报功能。
isReportLaunch	boolean	选填	false	是否开启APP启动时上报功能。

上报日志策略说明

LTSSDK支持四种日志上报策略：阈值上报、定时上报、应用切后台和启动时上报，根据场景选择合适的值配置cacheThreshold、timeInterval、isReportBackground、isReportLaunch等参数。

权限配置

LTSSDK 需要 Android 的网络权限，如果你没有配置，请将下面内容的写入APP 的 Manifest.xml 文件中：

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
...
// 加入下面的内容
<uses-permission android:name="android.permission.INTERNET" />
...
</manifest>
```

混淆配置

如App对代码进行乱序混淆，则在混淆配置文件中添加代码段：

```
-keep class com.cloud.**{*;}
```

接入调试

在开发过程中可以启用调试模式，借助控制台日志实时查看上报日志的记录情况，观察具体结果并根据需要进行调整。

需要通过调用LTSSDK的静态方法来修改，SDK支持Debug、Info、Warning 、 Error 和 Off 5个级别的 默认为静默输出（ Off ）。5个级别分别对应的参数：LogLevel.DEBUG、LogLevel.INFO、LogLevel.WARNING、LogLevel.ERROR、LogLevel.OFF。代码示例如下：

```
LTSSDK.setLogLevel(LogLevel.DEBUG)
```

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.8 云日志服务百度小程序 SDK

云日志服务百度小程序SDK提供了百度小程序上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务百度小程序SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**管理日志流**。

版本更新说明

表 4-144 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例上报。

使用说明

百度小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。

需将上报地址添加到百度开发者平台域名列表中，上报地址请参考**配置参数说明**。

安装 SDK

在项目根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

说明

您可以在**开源仓地址**下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
    onLaunch(options) {
        // 初始化
        const baidumini = new miniSDK({
            // 上报region
            region: string,
            // 华为云项目ID
            projectId: string,
            // 上报地址
            url: string,
            // LTS日志组ID
            groupId: string,
            // LTS日志流ID
            streamId: string,
            // 日志所属组
            group: string,
            // 调试日志等级
            debug: string,
            // 当前小程序所属平台
            platform: string,
            // 上报条数阈值
            cacheThreshold: number,
            // 上报时间阈值
            timeInterval: number,
        });
        // 立即上报单条带标签
        baidumini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
        // 立即上报单条 不带标签
        baidumini.reportImmediately([{    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }, { 'key': 'value' }]);
        // 缓存上报多条 带标签
        baidumini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
        // 缓存上报多条 不带标签
        baidumini.report([{ 'name': 'xiaohong', 'age': 18 }, {    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }]);
        // 缓存上报多条 带多个标签（最多50个）
        baidumini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'mini', 'link': '/', from: 'baidu'});
    }
});
```

配置参数说明

- 配置参数说明

表 4-145 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在百度小程序后台开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用
group	string	选填	-	日志组ID

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的数据。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.9 云日志服务微信小程序 SDK

云日志服务微信小程序SDK提供了微信小程序上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务微信小程序SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**日志流**。

版本更新说明

表 4-146 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例。

使用说明

- 微信小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。
- 为确保微信小程序日志上报正常，需将上报地址添加到微信开发者平台域名列表，上报地址请参见**配置参数说明**。

安装 SDK

在项目根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

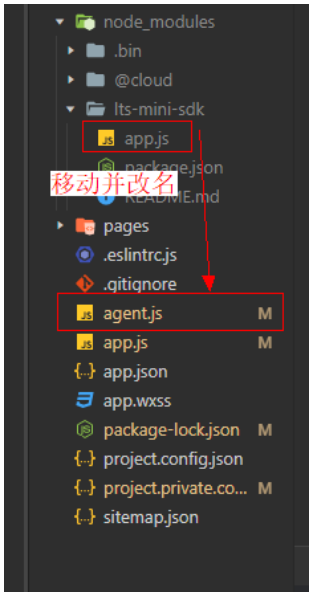
说明

您可以在**开源仓地址**下载最新的SDK。可以将软件包中的app.js复制到项目文件目录下使用或者项目开启npm，使用node_module下的文件。

集成 SDK

- 采用npm方式集成SDK。
  a. 确保项目有“package.json”文件，如果项目中没有“package.json”文件，可以在项目的根目录下，使用以下命令来创建：npm init。
  b. 运行安装SDK的命令，安装SDK软件包。
  c. 单击开发者工具菜单栏中的“工具 > 构建npm”，构建当前工程的npm库文件。

- 采用文件引入方式集成SDK。
  a. 运行安装SDK的命令，安装SDK软件包。
  b. 找到SDK文件夹中的app.js文件“node_modules > lts-mini-sdk > app.js”，将app.js文件从node_module复制到根路径并改名。



示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
    onLaunch(options) {
        // 初始化
        const wxmini = new miniSDK({
            // 上报region
            region: string,
```



```
        // 华为云项目ID
        projectId: string,
        // 上报地址
        url: string,
        // LTS日志组ID
        groupId: string,
        // LTS日志流ID
        streamId: string,
        // 调试日志等级
        debug: string,
        // 当前小程序所属平台
        platform: string,
        // 上报条数阈值
        cacheThreshold: number,
        // 上报时间阈值
        timeInterval: number,
    });
    // 立即上报单条带标签
    wxmini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
    // 立即上报单条 不带标签
    wxmini.reportImmediately([{   key: 'value',   number: 1,   array: [],   json: {     json: 'json'   } }, { 'key': 'value' }]);
    // 缓存上报多条 带标签
    wxmini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
    // 缓存上报多条 不带标签
    wxmini.report([{ 'name': 'xiaohong', 'age': 18 }, {   key: 'value',   number: 1,   array: [],   json: {     json: 'json'   } }]);
    // 缓存上报多条 带多个标签（最多50个）
    wxmini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'mini', 'link': '/', from: 'wx'});
  }
});
```

配置参数说明

- 配置参数说明

表 4-147 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。

url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在微信小程序后台开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3

区域名称	RegionName
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.10 云日志服务钉钉小程序 SDK

云日志服务小程序SDK提供了钉钉小程序上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务钉钉小程序SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**管理日志流**。

版本更新说明

表 4-148 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例上报。

使用说明

- 钉钉小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。
- 为确保钉钉小程序日志上报正常，需将上报地址添加到钉钉开发者平台域名列表，上报地址请参见**配置参数说明**。

安装 SDK

在项目根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

说明

您可以在**开源仓地址**下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
    onLaunch(options) {
        // 初始化
        const ddmini = new miniSDK({
            // 上报region
            region: string,
            // 华为云项目ID
            projectId: string,
            // 上报地址
            url: string,
            // LTS日志组ID
            groupId: string,
            // LTS日志流ID
            streamId: string,
            // 调试日志等级
            debug: string,
            // 当前小程序所属平台
            platform: 'dd',
            // 上报条数阈值
```

```
            cacheThreshold: number,
            // 上报时间阈值
            timeInterval: number,
        });
        // 立即上报单条带标签
        ddmini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
        // 立即上报单条 不带标签
        ddmini.reportImmediately([{    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }, { 'key': 'value' }]);
        // 缓存上报多条 带标签
        ddmini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
        // 缓存上报多条 不带标签
        ddmini.report([{ 'name': 'xiaohong', 'age': 18 }, {    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }]);
        // 缓存上报多条 带多个标签（最多50个）
        ddmini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'mini', 'link': '/', from: 'dd'});
    }
});
```

配置参数说明

- 配置参数说明

表 4-149 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	https://lts-access.cn-north-4.myhuaweicloud.com	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在钉钉小程序开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。
debug	boolean	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。

cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述

content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.11 云日志服务支付宝小程序 SDK

云日志服务小程序SDK提供了支付宝小程序上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务小程序SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**管理日志流**。

版本更新说明

表 4-150 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例上报。

使用说明

- 支付宝小程序SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请**提工单**申请。
- 为确保支付宝小程序日志上报正常，需将上报地址添加到支付宝开发者平台域名列表，上报地址请参见**配置参数说明**。

安装 SDK

在项目根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

说明

您可以在**开源仓地址**下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk').default;
// import miniSDK from 'lts-mini-sdk';
App({
    onLaunch(options) {
        // 初始化
        const mymini = new miniSDK({
            // 上报region
            region: string,
            // 华为云项目ID
            projectId: string,
            // 上报地址
            url: string,
            // LTS日志组ID
            groupId: string,
            // LTS日志流ID
            streamId: string,
            // 调试日志等级
            debug: string,
            // 当前小程序所属平台
            platform: 'my',
            // 上报条数阈值
            cacheThreshold: number,
            // 上报时间阈值
            timeInterval: number,
        });
        // 立即上报单条带标签
        mymini.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
        // 立即上报单条 不带标签
        mymini.reportImmediately([{    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }, { 'key': 'value' }]);
        // 缓存上报多条 带标签
        mymini.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value' });
        // 缓存上报多条 不带标签
        mymini.report([{ 'name': 'xiaohong', 'age': 18 }, {    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } }]);
        // 缓存上报多条 带多个标签（最多50个）
        mymini.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'mini', 'link': '/', from: 'zhifubao'});
    }
});
```

配置参数说明

- 配置参数说明

表 4-151 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。
streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。该域名地址需要在支付宝小程序开发设置服务器域名处（具体操作界面以小程序官网为准）注册为合法域名，128个字符上限。
debug	boolean	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu ： 百度小程序，wx ：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述

content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：登录云日志服务页面，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击“日志组名称”可查看日志流列表，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.12 云日志服务快应用 SDK

云日志服务快应用SDK提供了快应用上报日志的一系列方法，方便用户直接使用编码方式上报日志到云日志服务后台。

传输协议

HTTPS

使用前提

- 使用云日志服务快应用SDK前，您需要注册华为账号并开通华为云。
- 确认云日志服务的区域，请用户根据所在区域，选择region name。
- 获取华为云账号的项目ID（project id），步骤参考：请参见“我的凭证 > **API凭证**”。
- 获取需要上报到LTS的**日志组ID**和**日志流ID**。
- 日志流需要开启匿名写入功能。详细操作请参考**管理日志流**。

版本更新说明

表 4-152 版本更新说明

版本号	更新说明
1.0.24	支持更多region：华东-上海一、华南-广州。
1.0.23	新增支持快应用上报。
1.0.21	1. 废弃config方法，优先使用new SDK创建一个新的实例。 2. 去除代码中对三方包的依赖和存在的中文符号。
1.0.19	修改时间阈值的范围从1-60改为1-1800，其默认值从30改为3。
1.0.18	1. 调整日志级别等级。 2. 支持labels嵌套。
1.0.15	新增多实例上报。

使用说明

快应用SDK支持跨云/本地上报日志，当前仅支持华北-北京四、华东-上海一、华南-广州的白名单用户，如有需要请提工单申请。

安装 SDK

在项目根目录下通过运行“npm i lts-mini-sdk”命令，安装SDK软件包。

说明

您可以在**开源仓地址**下载最新的SDK。

示例代码

```
const miniSDK = require('lts-mini-sdk/quick-app').default;
// import miniSDK from 'lts-mini-sdk/quick-app';
App({
    onLaunch(options) {
        // 初始化
        const quickAppSdk = new miniSDK({
            // 上报region
            region: string,
            // 华为云项目ID
            projectId: string,
            // 上报地址
            url: string,
            // LTS日志组ID
            groupId: string,
            // LTS日志流ID
            streamId: string,
            // 日志所属组
            group: string,
            // 调试日志等级
            debug: string,
            // 当前小程序所属平台
            platform: string,
            // 上报条数阈值
            cacheThreshold: number,
            // 上报时间阈值
            timeInterval: number,
        });
        // 立即上报单条带标签
        quickAppSdk.reportImmediately({ 'name': 'xiaoming', 'age': 18 }, { 'key': 'value' });
        // 立即上报单条 不带标签
        quickAppSdk.reportImmediately([{    key: 'value',    number: 1,    array: [],    json: {     json: 'json'    } },
{ 'key': 'value' }]);
        // 缓存上报多条 带标签
        quickAppSdk.report([{ 'name': 'xiaohong', 'age': 18 }, { 'name': 'xiaobai', 'age': 20 }], { 'key': 'value });
        // 缓存上报多条 不带标签
        quickAppSdk.report([{ 'name': 'xiaohong', 'age': 18 }, {    key: 'value',    number: 1,    array: [],    json:
{     json: 'json'    } }]);
        // 缓存上报多条 带多个标签（最多50个）
        quickAppSdk.report([{ 'name': 'xiaohong', 'name': 'xiaolan' }], {'version': '1.0.0', 'render': 'mini', 'link': '/',
from: 'baidu'});
    }
});
```

配置参数说明

- 配置参数说明

表 4-153 配置参数说明

字段	类型	是否必填	默认值	描述
region	string	必填	-	上报LTS所处的region。
projectId	string	必填	-	账号的项目ID，128个字符上限。
groupId	string	必填	-	LTS日志组ID，128个字符上限。

streamId	string	必填	-	LTS日志流ID，128个字符上限。
url	string	选填	-	用于上报的公网地址域名，支持带端口号，比如：https://lts-access.cn-north-4.myhuaweicloud.com:443；如未设置url，将根据region自动生成链接，格式如下：https://lts-access.{region}.myhuaweicloud.com。
debug	string	选填	OFF	控制台调试信息的输出等级，有OFF、ERROR、WARN、INFO、DEBUG五个等级，填入无效值会默认为OFF，最高等级为DEBUG，层级往下递减。值为true时开启DEBUG等级的日志，值为false时日志等级为OFF。
cacheThreshold	int	选填	30	默认30条上报条数阈值，当缓存到达阈值时上报缓存中的数据 30 <= cacheThreshold <= 1000
timeInterval	number	选填	3	默认3s 上报阈值时间，当道道阈值时间上报缓存中的数据 1 <= timeInterval <= 1800
platform	string	选填	baidu	支持上报的平台。目前支持以下平台：baidu：百度小程序，wx：微信小程序，dd：钉钉小程序，my：支付宝小程序，quick：快应用
group	string	选填	-	日志组ID

- 日志上报report方法参数说明

字段	类型	是否必填	默认值	描述

content	Object | Array<Object>	必填	-	需要上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 **说明** 用户的日志字段名称不允许包含双下划线（__）。
labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

- 日志立即上报reportImmediately方法参数说明

字段	类型	是否必填	默认值	描述
content	Object | Array<Object>	选填	-	立即上报的日志对象或者对象数组，限制：Object：最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。 array<Object>：数组中的每条Object数据最多300个键值对，转成JSON字符串最大支持长度1024*30个字节，超出部分将被截断。不填入内容时，将立即上报缓存中的数据。不填入内容时，将立即上报缓存中的日志。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

labels	Object	选填	空	日志标签，最外层键值对50个以内，最外层key最大长度为64个字符，最外层key值只能由字母开头，字母、数字或下划线组成，标签转成JSON字符串最大支持长度1024*30个字符，超出限制将不上报此条信息。 **说明** 用户的日志字段名称不允许包含双下划线（__）。

参数获取方式

- 区域表

区域名称	RegionName
华北-北京四	cn-north-4
华东-上海一	cn-east-3
华南-广州	cn-south-1

- 日志组ID：在云日志服务控制台，选择“日志管理”，鼠标悬浮在日志组名称上，可查看日志组名称和日志组ID。
- 日志流ID：单击日志组名称对应的 按钮，鼠标悬浮在日志流名称上，可查看日志流名称和日志流ID。

4.5.13 LTS SDK 接入错误码

当您使用LTS SDK报错时，请参见表4-154进行处理。

表 4-154 错误码

状态码	错误码	错误信息	描述	处理措施
参数类错误（LTS.00XX）	LTS.0001	%@ is null.	参数%@为空。	请设置非空的参数后重试。
	LTS.0002	%@ is invalid.	参数%@的类型、格式无效。	请设置有效的参数后重试。
	LTS.0003	The length of %@ exceeds the maximum value of %@.	参数%@的长度超过了最大值%@。	请修改参数长度后重试。
	LTS.0004	The value of %@ must be between %@ and %@.	参数%@的值必须在%@-%@之间。	请修改参数值后重试。

状态码	错误码	错误信息	描述	处理措施
	LTS.0006	%@ doesn't match pattern.	参数%@不匹配正则表达式。	请设置符合正则表达式的参数后重试。
	LTS.0007	Invalid configuration parameters.	无效的配置参数。	请设置正确的配置项参数后重试。
	LTS.0009	Unsupported region.	上报日志不支持该地区。	请使用下列有效区域：cn-north-4。
集成类错误（LTS.01XX）	LTS.0100	The LTSSDK is not initialized.	SDK未初始化。	请设置符合规则的配置参数，正确初始化SDK。
	LTS.0102	The LTSSDK requires %@ or higher.	SDK要求%@及以上操作系统。	请根据提示调整最低操作系统版本。
	LTS.0103	Open database failed, code=%@.	打开数据库失败。	请根据sqlite3 error code查询具体情况。
	LTS.0104	Create database table failed, code=%@, error=%@.	创建数据表失败。	请根据sqlite3 error code查询具体情况。
	LTS.0105	Insert into database failed, code=%@, error=%@.	插入数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0106	Delete from database failed, code=%@, error=%@.	删除数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0107	Select from database failed, code=%@, error=%@.	查询数据失败。	请根据sqlite3 error code查询具体情况。
	LTS.0108	Close database failed, code=%@.	关闭数据库失败。	请根据sqlite3 error code查询具体情况。

状态码	错误码	错误信息	描述	处理措施
	LTS.0109	Reduce database size failed, code=%@, error=%@.	减小数据库大小失败。	请根据sqlite3 error code查询具体情况.
	LTS.0110	Begin transaction failed, code=%@, error=%@.	开启事务失败。	请根据sqlite3 error code查询具体情况。
	LTS.0111	Commit transaction failed, code=%@, error=%@.	提交事务失败。	请根据sqlite3 error code查询具体情况。
	LTS.0112	The count of database records exceeds the maximum value of %@, will delete %@ records before inserting.	数据库达到最大日志条数限制。	将自动删除适量老日志，再插入新日志。
	LTS.0113	Add column(%@) to database failed, code=%d, error=%s.	添加数据库表字段失败	请根据sqlite3 error code查询具体情况。
	LTS.0114	Update column(%@) to database failed, code=%d, error=%s.	更新数据库表字段失败	请根据sqlite3 error code查询具体情况。
系统类错误（LTS.02XX）	LTS.0201	Request signature failed.	网络请求签名失败。	请检查region和设备时间（时区）是否正确。
	LTS.0202	Write to file failed.	写入文件失败。	请检查文件写权限。
	LTS.0203	No file read and write permissions.	无文件读写权限。	请授权文件/目录读写权限。

状态码	错误码	错误信息	描述	处理措施
网络类错误（LTS.03XX）	LTS.0300	Request failed，response code=%@，error=%@.	请求发送失败。	请参考API参考中的错误码。

4.6 跨 IAM 账号接入 LTS

当您选择了跨账号接入-日志流映射方式时，通过创建委托，您可以将委托账号的日志流映射到被委托方账号，即就是将委托账号的日志流映射到当前云日志服务账号的日志流下。

说明

跨账号接入成功后，如果账号A在IAM上删除委托，则云日志服务无法感知到该委托被删除，配置的跨账号接入依然生效；如果您不再使用跨账号接入功能，可直接通知账号B删除接入配置。

前提条件

已创建委托关系。

限制条件

数据未同步完成前，目标日志流数据与源日志流可能会有一定偏差。建议1小时后，查看接入数据。

设置跨账号接入

日志服务接入方式选择跨账号接入时，按照如下操作完成接入配置。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志接入”，单击“跨账号接入-日志流映射”进行跨账号接入配置。

步骤3 或者在左侧导航栏中，选择“日志管理”，单击目标日志流的名称进入日志详情页面，单击右上角  ，在弹出页面中，选择“日志接入”页签，单击“接入日志”，在新打开的页面，选择“跨账号接入-日志流映射”进行跨账号接入配置。

步骤4 选择委托。

配置相关参数，请参见表4-155，完成后，单击“下一步：日志流映射”。

表 4-155 委托参数配置

参数	说明
委托名称	填写委托人在IAM中创建的委托名称。委托人账号可通过创建委托将资源管理权限委托给其他华为云账号。

参数	说明
委托人账号名称	填写委托人账号名称，以验证委托关系。

步骤5 日志流映射。

在日志流映射页面，配置接入规则，有两种方式：自动配置和手动配置。

- **自动配置**
  a. 在日志流映射页面，单击“自动配置”。
  b. 在弹出的自动配置页面中，配置相关参数信息，完成后，单击“确定”。

表 4-156 自动配置接入规则

参数	说明
规则名称前缀	填写规则名称前缀，自动配置将使用您配置的规则名称前缀，产生多条接入规则。 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。可不填写，默认规则名称前缀为rule。
从委托账号中选择您希望接入的日志组/日志流	选择希望接入的日志组/日志流，最多支持选择20条。

说明

通过自动配置的接入规则，被委托方中的目标日志组、目标日志流名称默认同委托方中源日志组、源日志流名称保持一致，也支持手动修改。

  c. 单击“预览”，查看预览结果。

说明

1. 预览结果有两种：
   - **将创建新的目标日志流**：被委托方中新建的目标日志组/日志流。
   - **接入已存在的目标日志流**：被委托方中已存在的目标日志组/日志流。
2. 预览报错情况如下：
   - 源日志流xxx，已配置为目标日志流
   - 目标日志流xxx，已配置为源日志流
   - 目标日志流xxx，已存在于其它日志组
   - 目标日志流xxx，存在于不同目标日志组
   - 规则名称重复
   - 源日志流xxx，已存在映射关系
   - 日志组/日志流超过最大创建条数

   当提示以上报错时，须删除日志流对应的接入规则。

d. 预览完成后，单击“提交”。

- **手动配置**

a. 在日志流映射页面，单击“添加规则”。

表 4-157

参数		说明
规则名称		默认为rule_xxx，也可根据您的需要进行自主命名。 只支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。
委托方	源日志组	委托方的日志组，在原有的日志组中进行选择。
	源日志流	委托方的日志流，在原有的日志流中进行选择。
被委托方	目标日志组	被委托方的日志组，可在原有的日志组中进行选择或直接输入名称进行新建日志组。
	目标日志流	被委托方的日志流，可在原有的日志流中进行选择或直接输入名称进行新建日志流。

b. 单击“预览”，查看预览结果。

说明

1. 预览结果有两种：
   - **将创建新的目标日志流**：被委托方中新建的目标日志组/日志流。
   - **接入已存在的目标日志流**：被委托方中已存在的目标日志组/日志流。
2. 预览报错情况有五种：
   - 源日志流xxx，已配置为目标日志流。
   - 目标日志流xxx，已配置为源日志流。
   - 目标日志流xxx，已存在于其它日志组。
   - 目标日志流xxx，存在于不同目标日志组。
   - 规则名称重复。
   - 源日志流xxx，已存在映射关系。
   - 日志组/日志流超过最大创建条数。

   当提示以上报错时，须删除日志流对应的接入规则。

c. 预览完成后，单击“提交”，等待创建日志接入成功。

步骤6 完成。

说明

配置完成后，数据将会在1小时内完成同步，请您耐心等待。

- 当接入多个日志流时，单击“返回接入配置列表”可查看日志接入列表。

- 当接入单个日志流时，单击“返回接入配置列表”可查看日志接入列表；单击“查看日志流”可查看已接入的日志详情。

----结束

4.7 使用 KAFKA 协议上报日志到 LTS

您可以通过Kafka协议上报日志到日志服务，目前支持各类Kafka Producer SDK或采集工具，仅依赖于Kafka协议。支持以下场景：

- **场景1**：您已有基于开源采集的自建系统，仅修改配置文件便可以将日志上报到LTS，例如Logstash。
- **场景2**：您希望通过 Kafka producer SDK 来采集日志并上报，不必再安装采集ICAgent。

说明

目前此功能仅支持华北-北京四、华南-广州、华东-上海二、亚太-新加坡，其他局点需要**提交工单**申请使用。

前提条件

- 使用云日志SDK前，您需要注册用户账号，并开通云日志服务。
- 确认云日志服务的区域，请用户根据所在区域，获取regionid。
- **获取华为账号的AK/SK**。
- 获取华为云账号的项目ID（ project id ），步骤参考：请参见“我的凭证 >**API凭证**”。
- 获取需要上报到LTS的日志组ID、日志流ID。
- 当前仅支持内网上报，需要在ECS主机上使用。

相关限制

- 当前仅支持内网上报，端口固定为9095，IP根据所在局点进行配置。
- 支持 Kafka 协议版本为：1.0.X，2.X.X，3.X.X。
- 支持压缩方式：gzip，snappy，lz4。
- KAFKA认证方式为 SASL_PLAINTEXT 认证。
- KAFKA协议的ACKS参数必须设置为0。

配置方式

- 使用Kafka协议上报日志时，需要使用到的通用参数如下。

表 4-158 通用参数

参数名称	描述	类型
projectId	用户账号的项目ID（ project id ）	String
logGroupId	LTS的日志组ID	String
logStreamId	LTS的日志流ID	String

参数名称	描述	类型
regionName	云日志服务的区域	String
accessKey	用户账号的AK	String
accessSecret	用户账号的SK	String

- 使用Kafka协议上报日志时，需要配置以下参数。

表 4-159 配置参数

参数名称	说明
连接类型	当前支持SASL_PLAINTEXT
hosts	Kafka的IP和PORT地址，格式为lts-kafka.${regionName}.${external_global_domain_name}:9095或lts-access.${regionName}.${external_global_domain_name}:9095 其中IP根据局点进行配置，PORT固定为9095。详细请参考**参数获取方式**，例如北京四局点对应hosts为 lts-kafka.cn-north-4.myhuaweicloud.com:9095。
topic	Kafka的topic名称，格式为 ${日志组ID}_${日志流ID}，即LTS的日志组ID和日志流ID通过下划线连接，作为topic的名称。
username	Kafka访问用户名，配置为用户账号的项目ID。
password	Kafka访问密码，格式为${accessKey}#${accessSecret}，即用户账号的AK和SK通过#连接，作为Kafka的访问密码。
headers	设置自定义label字段时，需要配置headers。 headers中添加header，key值为LTS_LOG_TYPE，value值为FORMAT，用户需要上报符合要求的规范化日志。

- ${message}日志格式

  仅当headers中添加了key为LTS_LOG_TYPE，value为FORMAT的header时，日志需要符合该格式规范。

表 4-160 日志参数

参数名称	是否必选	参数类型	描述
tenant_project_id	是	String	用户账号的项目ID。
tenant_group_id	是	String	LTS的日志组ID。
tenant_stream_id	是	String	LTS的日志流ID。

参数名称	是否必选	参数类型	描述
log_time_ns	是	Long	日志数据采集时间，UTC时间（纳秒）。 **说明** 采集时间需在日志存储时间范围之内，否则上报日志会被删除。比如日志组的日志存储时间是7天，则此参数不应早于当前时间的7天前。
contents	是	Array of String	日志内容
labels	是	Object	用户自定义label。 **说明** 请不要将字段名称设置为**内置保留字段**，否则可能会造成字段名称重复、查询不精确等问题。

日志示例

```
{
   "tenant_project_id": "${projectId}",
   "tenant_group_id": "${logGroupId}",
   "tenant_stream_id": "${logStreamId}",
   "log_time_ns": "XXXXXXXXXXXXXXXXXX",
   "contents": [
      "This is a log 1",
      "This is a log 2"
   ],
   "labels": {
      "type": "kafka"
   }
}
```

调用示例

1. Beat系列软件调用（FileBeat等）。以FileBeat为例，配置参数如下：

```
output.kafka:
hosts: ["${ip}:${port}"]
partition.round_robin:
reachable_only: false
username: "${projectId}"
password: "${accessKey}#${accessSecret}"
topic: "${logGroupId}_${logStreamId}"
sasl.mechanism: "PLAIN"
security.protocol: "SASL_PLAINTEXT"
acks: "0"
compression: gzip
```

2. 通过Logstash软件上报日志。

```
input {
stdin {}
}
output {
kafka {
# 配置地址
bootstrap_servers => "${ip}:${port}"
# 配置topic
topic_id => "${logGroupId}_${logStreamId}"
# 配置消息确认机制
acks => "0"
```



```
# 配置压缩方式
compression_type => "gzip"
# 配置认证方式
security_protocol => "SASL_PLAINTEXT"
sasl_mechanism => "PLAIN"
# 用户名 projectId 密码 accessKey#accessSecret
sasl_jaas_config => "org.apache.kafka.common.security.plain.PlainLoginModule required username='$
{projectId}' password='${accessKey}#${accessSecret}';"
}
}
```

3. 通过Flume软件上报日志。

```
#Name
a1.sources = r1
a1.channels = c1
a1.sinks = k1
#Source
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /tmp/test.txt
a1.sources.r1.fileHeader = true
a1.sources.r1.maxBatchCount = 1000
#Channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100
#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";
#Bind
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

SDK 调用示例

1. Java SDK调用示例。

   maven依赖（示例kafka协议版本为2.7.1）：

```
<dependencies>
    <dependency>
        <groupId>org.apache.kafka</groupId>
        <artifactId>kafka-clients</artifactId>
        <version>2.7.1</version>
    </dependency>
</dependencies>
```

   代码示例：

```
package org.example;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.Producer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;
public class ProducerDemo {
public static void main(String[] args) {
Properties props = new Properties();
// 配置地址
props.put("bootstrap.servers", "${ip}:${port}");
// 配置消息确认机制
props.put("acks", "0");
// 配置认证方式
props.put("security.protocol", "SASL_PLAINTEXT");
```

```
props.put("sasl.mechanism", "PLAIN");
// 用户名 projectId 密码 accessKey#accessSecret
props.put("sasl.jaas.config",
"org.apache.kafka.common.security.plain.PlainLoginModule required username='${projectId}'
password='${accessKey}#${accessSecret}';");
// 配置压缩方式
props.put("compression.type", "${compress_type}");
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");
// 1.创建一个生产者对象
Producer<String, String> producer = new KafkaProducer<>(props);
// 2.调用send方法
for (int i = 0; i < 1; i++) {
ProducerRecord record = new ProducerRecord<>("${logGroupId}_${logStreamId}", "${message}");
// 配置recordHeader
// record.headers().add(new RecordHeader("LTS_LOG_TYPE","FORMAT".getBytes()));
producer.send(record);
}
// 3.关闭生产者
producer.close();
}
}
```

2. Python SDK调用示例。

```
from kafka import KafkaProducer
producer = KafkaProducer(
# 配置地址
bootstrap_servers="${ip}:${port}",
# 配置消息确认机制
acks="0",
# 配置压缩方式
compression_type ="${compression_type}"
# 配置认证方式
sasl_mechanism="PLAIN",
security_protocol="SASL_PLAINTEXT",
# 用户名 projectId 密码 accessKey#accessSecret
sasl_plain_username="${projectId}",
sasl_plain_password="${accessKey}#${accessSecret}"
)
print('start producer')
for i in range(0, 3):
    data = bytes("${message}", encoding="utf-8")
    future = producer.send("${logGroupId}_{logStreamId}", data)
    result = future.get(timeout=10)
    print(result)
print('end producer')
```

报错说明

当参数错误或不匹配时，会有相应的报错提示。

表 4-161 报错说明

报错信息	报错原因
TopicAuthorizationException	projectId（项目ID）、accessKey（AK）、accessSecret（SK）参数错误或者不匹配。
UnknownTopicOrPartitionException	logGroupId（日志组ID）、logStreamId（日志流ID）参数错误或者不匹配。

报错信息	报错原因
InvalidRecordException	仅当配置headers，上报规范化日志时，会出现此类报错： 日志格式错误或者日志中的projectId（项目ID）、logGroupId（日志组ID）、logStreamId（日志流ID）与外部设置参数不一致。

参数获取方式

表 4-162 区域表

区域名称	RegionName
华北-北京四	lts-kafka.cn-north-4.myhuaweicloud.com
华东-上海二	lts-kafka.cn-east-2.myhuaweicloud.com
华南-广州	lts-access.cn-south-1.myhuaweicloud.com
亚太-新加坡	lts-kafka.ap-southeast-3.myhuaweicloud.com

4.8 使用 Flume 采集器上报日志到 LTS

Flume是一个高可用的，高可靠的，分布式的海量日志采集、聚合和传输的系统，Flume支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume提供对数据进行简单处理，并写到各种数据接受方的能力。

用户使用Flume系统采集日志，并且通过LTS侧提供的KAFKA协议方式上报日志。以下是部分常用数据采集场景示例：

1. **使用Flume采集文本日志上报到LTS**
2. **使用Flume采集数据库表数据并且上报至LTS**
3. **使用Flume采集syslog协议传输的日志上报到LTS**
4. **通过Flume采集TCP/UDP协议传输的日志上报到LTS**
5. **通过Flume采集SNMP协议上报的设备管理数据并发送到LTS**
6. **使用默认拦截器处理日志**
7. **自定义拦截器处理日志**
8. **使用外部数据源丰富日志内容并上报到LTS**

前提条件

- 用户机器已经安装了JDK。
- 用户已经安装Flume，并且需要在Flume中配置文件中配置JDK路径。

使用 Flume 采集文本日志上报到 LTS

支持使用Flume采集文本日志内容上报至LTS，参考如下示例添加采集文本日志的conf文件。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

```
#Named
a1.sources = r1
a1.channels = c1
a1.sinks = k1

#Source
a1.sources.r1.type = TAILDIR
a1.sources.r1.channels = c1
a1.sources.r1.filegroups = f1
a1.sources.r1.filegroups.f1 = /tmp/test.txt
a1.sources.r1.fileHeader = true
a1.sources.r1.maxBatchCount = 1000

#Channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 100

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";

#Bind
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

使用 Flume 采集数据库表数据并且上报至 LTS

使用Flume采集数据库表数据并且上报至LTS，实现对表数据变动监控。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

步骤1 在**https://github.com/keedio/flume-ng-sql-source**页面下载flume-ng-sql-source插件，转换为jar包并取名为flume-ng-sql-source.jar，打包前注意将pom文件中的flume-ng-core 版本与flume安装版本保持一致，并且将jar包放在安装Flume包路径的lib目录下面，例如**FLUME_HOME/lib**目录下（例子中的**FLUME_HOME**为Flume安装路径，仅供参考，请以实际安装路径为准）。

步骤2 添加MySQL驱动到**FLUME_HOME/lib**目录下：

```
1、wget https://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.35.tar.gz
2、tar xzf mysql-connector-java-5.1.35.tar.gz
3、cp mysql-connector-java-5.1.35-bin.jar  FLUME_HOME/lib/
```

步骤3 添加采集MySQL的conf文件。

```
# a1表示agent的名称
# source是a1的输入源
# channels是缓冲区
# sinks是a1输出目的地，本例子sinks使用了kafka
a1.channels = c1
a1.sources = r1
a1.sinks = k1

#source
a1.sources.r1.type = org.keedio.flume.source.SQLSource
```



```
# 连接mysql的一系列操作，{mysql_host}改为你虚拟机的ip地址，可以通过ifconfig或者ip addr查看，
{database_name}改为数据库名称
# url中要加入?useUnicode=true&characterEncoding=utf-8&useSSL=false，否则有可能连接失败
a1.sources.r1.hibernate.connection.url = jdbc:mysql://{mysql_host}:3306/{database_name}?
useUnicode=true&characterEncoding=utf-8&useSSL=false
# Hibernate Database connection properties
# mysql账号，一般都是root
a1.sources.r1.hibernate.connection.user = root
# 填入你的mysql密码
a1.sources.r1.hibernate.connection.password = xxxxxxxx
a1.sources.r1.hibernate.connection.autocommit = true
# mysql驱动
a1.sources.r1.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect
a1.sources.r1.hibernate.connection.driver_class = com.mysql.jdbc.Driver
# 存放status文件
a1.sources.r1.status.file.path = FLUME_HOME/bin
a1.sources.r1.status.file.name = sqlSource.status
# Custom query
# 填写需要采集的数据表名{table_name}，也可以使用下面的方法：
a1.sources.r1.custom.query = select * from {table_name}


#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required username="${projectId}" password="${accessKey}#${accessSecret}";


a1.channels.c1.type = memory
a1.channels.c1.capacity = 10000
a1.channels.c1.transactionCapacity = 10000
a1.channels.c1.byteCapacityBufferPercentage = 20
a1.channels.c1.byteCapacity = 800000
```

步骤4 启动Flume后，即可开始采集数据库中的表数据到LTS。

----结束

使用 Flume 采集 syslog 协议传输的日志上报到 LTS

Syslog协议是一种用于在IP网络中传输日志消息的协议，通过Flume将syslog协议传输的日志采集并上报到LTS。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

- 接收UDP日志，参考如下示例添加采集Syslog协议的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type=syslogudp
#host_port为syslog服务器的端口
a1.sources.r1.port = {host_port}
#host_ip为syslog服务器的ip地址
a1.sources.r1.host = {host_ip}
a1.sources.r1.channels = c1

a1.channels.c1.type = memory

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
```



```
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";
a1.sinks.k1.channel = c1
```

- 接收TCP日志，参考如下示例添加采集Syslog协议的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type=syslogtcp
#host_port为syslog服务器的端口
a1.sources.r1.port = {host_port}
#host_ip为syslog服务器的ip地址
a1.sources.r1.host = {host_ip}
a1.sources.r1.channels = c1

a1.channels.c1.type = memory

#Sink
a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";
a1.sinks.k1.channel = c1
```

通过 Flume 采集 TCP/UDP 协议传输的日志上报到 LTS

通过Flume采集TCP/UDP协议传输的日志上报到LTS。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

- 采集TCP端口日志，参考如下示例添加采集端口的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcat
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = {host_port}

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

- 采集UDP端口日志，参考如下示例添加采集端口的conf文件。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1
```



```
a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = {host_port}

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

通过 Flume 采集 SNMP 协议上报的设备管理数据并发送到 LTS

通过Flume采集SNMP协议上报的设备管理数据并发送到LTS。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

- 监听SNMP协议通信端口号161。参考如下示例添加SNMP协议接受日志的conf。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1

a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 161

a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

- 监听SNMP协议陷阱(Trap)通信的端口号162，参考如下示例添加SNMP协议接受日志的conf。

```
a1.sources = r1
a1.sinks = k1
a1.channels = c1


a1.sources.r1.type = netcatudp
a1.sources.r1.bind = 0.0.0.0
a1.sources.r1.port = 162


a1.sinks.k1.type = org.apache.flume.sink.kafka.KafkaSink
a1.sinks.k1.kafka.topic = ${logGroupId}_${logStreamId}
a1.sinks.k1.kafka.bootstrap.servers = ${ip}:${port}
a1.sinks.k1.kafka.producer.acks = 0
```



```
a1.sinks.k1.kafka.producer.security.protocol = SASL_PLAINTEXT
a1.sinks.k1.kafka.producer.sasl.mechanism = PLAIN
a1.sinks.k1.kafka.producer.compression.type = gzip
a1.sinks.k1.kafka.producer.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="${projectId}" password="${accessKey}#${accessSecret}";

a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

使用默认拦截器处理日志

使用Flume采集器时，拦截器是简单的插件式组件，设置在Source和Channel之间。Source接收到的事件Event，在写入Channel之前，拦截器都可以进行转换或者删除这些事件。每个拦截器只处理同一个Source接收到的事件。介绍使用默认拦截器处理日志。

- 时间戳拦截器

  该拦截器的作用是将时间戳插入到flume的事件报头中。如果不使用任何拦截器，flume接受到的只有message。时间戳拦截器的配置。 参数默认值描述type，类型名称timestamp，也可以使用类名的全路径preserveExisting为false。如果设置为true，若事件中报头已经存在，不会替换时间戳报头的值。source连接到时间戳拦截器的配置：

  ```
  a1.sources.r1.interceptors = timestamp
  a1.sources.r1.interceptors.timestamp.type=timestamp
  a1.sources.r1.interceptors.timestamp.preserveExisting=false
  ```

- 正则过滤拦截器

  在日志采集的时候，可能有一些数据是不需要的，添加过滤拦截器可以过滤掉不需要的日志，也可以根据需要收集满足正则条件的日志。参数默认值描述type，类型名称REGEX_FILTER。excludeEvents为false时默认收集匹配到的事件。如果为true，则会删除匹配到的event，收集未匹配到的。source连接到正则过滤拦截器的配置：

  ```
  a1.sources.r1.interceptors = regex
  a1.sources.r1.interceptors.regex.type=REGEX_FILTER
  a1.sources.r1.interceptors.regex.regex=(today)|(Monday)
  a1.sources.r1.interceptors.regex.excludeEvents=false
  ```

  这样配置的拦截器就只会接收日志消息中带有today或者Monday的日志。

- 搜索并替换拦截器

  拦截器基于Java正则表达式提供简单的基于字符串的搜索和替换功能。配置如下：

  ```
  # 拦截器别名
  a1.sources.r1.interceptors = search-replace
  # 拦截器类型，必须是search_replace
  a1.sources.r1.interceptors.search-replace.type = search_replace

  #删除事件正文中的字符，根据正则匹配event内容
  a1.sources.r1.interceptors.search-replace.searchPattern = today
  # 替换匹配到的event内容
  a1.sources.r1.interceptors.search-replace.replaceString = yesterday
  # 设置字符集，默认是utf8
  a1.sources.r1.interceptors.search-replace.charset = utf8
  ```


自定义拦截器处理日志

在Flume中自定义拦截器的方式主要流程如下（以java语言为例），以下示例中的FLUME_HOME表示Flume的安装路径，例如/tools/flume（仅供参考），实际配置的时候，请以用户安装Flume的实际路径为准。

步骤1 创建MAVEN工程项目，引入Flume依赖。

根据集群中的 Flume 版本，引入 Flume 依赖，如下所示：

```
<dependencies>
    <dependency>
        <groupId>org.apache.flume</groupId>
        <artifactId>flume-ng-core</artifactId>
        <version>1.10.1</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

无需将该依赖打包进最后的JAR包中，故将其作用域设置为provided。

步骤2 创建类实现拦截器接口Interceptor，并且实现相关方法。

说明

- initialize() 方法：初始化拦截器操作，读取配置信息、建立连接等。
- intercept(Event event) 方法：用于拦截单个事件，并对事件进行处理。接收一个事件对象作为输入，并返回一个修改后的事件对象。
- intercept(List<Event> list) 方法：事件批处理，拦截事件列表，并对事件列表进行处理。
- close() 方法：关闭拦截器，在这里释放资源、关闭连接等。

```
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class TestInterceptor implements Interceptor {
    @Override
    public void initialize() {

    }
    @Override
    public Event intercept(Event event) {

        // 获取事件数据
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);
        // 检查事件数据中是否包含指定字符串
        if (eventData.contains("hello")) {
            // 如果包含指定字符串，则过滤掉该事件，返回 null
            return null;
        }

        return event;
    }

    @Override
    public List<Event> intercept(List<Event> events) {
        // 创建一个新的列表，存储处理过后的事件
        List<Event> interceptedEvents = new ArrayList<>();
        for (Event event : events) {
            Event interceptedEvent = intercept(event);
            if (interceptedEvent != null) {
                interceptedEvents.add(interceptedEvent);
            }
```

```
        }
        return interceptedEvents;
    }

    @Override
    public void close() {

    }

}
```

步骤3 构建拦截器，拦截器的创建和配置通常是通过 Builder 模式来完成的，完整的代码如下所示：

```
import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class TestInterceptor implements Interceptor {
    @Override
    public void initialize() {
    }
    @Override
    public Event intercept(Event event) {
        // 获取事件数据
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);
        // 检查事件数据中是否包含指定字符串
        if (eventData.contains("hello")) {
            // 如果包含指定字符串，则过滤掉该事件，返回 null
            return null;
        }
        return event;
    }
    @Override
    public List<Event> intercept(List<Event> events) {
        List<Event> interceptedEvents = new ArrayList<>();
        for (Event event : events) {
            Event interceptedEvent = intercept(event);
            if (interceptedEvent != null) {
                interceptedEvents.add(interceptedEvent);
            }
        }
        return interceptedEvents;
    }

    @Override
    public void close() {

    }

    // 拦截器构建
    public static class Builder implements Interceptor.Builder {

        @Override
        public void configure(Context context) {

        }

        @Override
        public Interceptor build() {
            return new TestInterceptor();
        }

    }

}
```


步骤4 转换为jar包，并且将其上传至Flume安装路径下的lib文件夹下（请以用户安装Flume的实际路径为准）。

步骤5 编写配置文件，需要将自定义的拦截器配置进去。

拦截器全类名配置时需要注意，格式为拦截器的全类名 + $Builder。

```
# 拦截器配置
# 拦截器定义
a1.sources.r1.interceptors = i1
# 拦截器全类名
a1.sources.r1.interceptors.i1.type = TestInterceptor$Builder
```

步骤6 运行Flume即可。

----结束

KV解析日志：用空格分隔字符串并且转换为Map类型字符串。

```
public class TestInterceptor implements Interceptor {
    @Override
    public void initialize() {
    }

    @Override
    public Event intercept(Event event) {
        // 获取事件数据
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);
        Map<String, Object> splitMap = new HashMap<>();
        String[] splitList = eventData.split(" ");
        for (int i = 0; i < splitList.length; i++) {
            splitMap.put("field" + i, splitList[i].trim());
        }
        eventData.setBody(splitMap.toString().getBytes(StandardCharsets.UTF_8));
        return event;
    }

    @Override
    public List<Event> intercept(List<Event> events) {
        List<Event> interceptedEvents = new ArrayList<>();
        for (Event event : events) {
            Event interceptedEvent = intercept(event);
            if (interceptedEvent != null) {
                interceptedEvents.add(interceptedEvent);
            }
        }
        return interceptedEvents;
    }

    @Override
    public void close() {
    }
}
```

使用外部数据源丰富日志内容并上报到 LTS

Flume数据传输的基本单元，以Event的形式将数据从源头传输至目的地。Event由Header和Body两部分组成，Header用来存放该Event的一些属性，为K-V结构，Body用来存放该条数据，形式为字节数组。

有外部数据源时，如果你需要丰富日志内容，例如修改日志内容、添加字段、删除内容等操作，将内容添加至Event的body中，Flume才能将日志内容上报到LTS。例如使用Java自定义扩展日志内容。以下示例中的参数介绍请参考**使用KAFKA协议上报日志**。

```
import com.alibaba.fastjson.JSONObject;
```



```
import org.apache.flume.Context;
import org.apache.flume.Event;
import org.apache.flume.interceptor.Interceptor;

import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

public class TestInterceptor implements Interceptor {
    @Override
    public void initialize() {
    }

    @Override
    public Event intercept(Event event) {
        // 获取事件数据，将原数据转换为json字符串并且添加额外字段
        String eventData = new String(event.getBody(), StandardCharsets.UTF_8);
        JSONObject object = new JSONObject();
        object.put("content", eventData);
        object.put("workLoadType", "RelipcaSet");
        eventData = object.toJSONString();
        eventData.setBody(eventData.getBytes(StandardCharsets.UTF_8));
        return event;
    }

    @Override
    public List<Event> intercept(List<Event> events) {
        List<Event> interceptedEvents = new ArrayList<>();
        for (Event event : events) {
            Event interceptedEvent = intercept(event);
            if (interceptedEvent != null) {
                interceptedEvents.add(interceptedEvent);
            }
        }
        return interceptedEvents;
    }

    @Override
    public void close() {
    }
}
```

5 日志搜索与分析（默认推荐）

5.1 日志搜索与分析概述

日志接入成功后，云日志服务支持对采集成功的日志数据进行搜索与分析。日志搜索与分析是运维中不可或缺的一环。通过合理的日志收集、高效的搜索方法和专业的分析工具，可以实现对系统或应用的全面监控和精细化管理。

- 执行搜索与分析前，需要将上报的日志进行结构化配置和索引配置，因为结构化后数据具有严格的长度和格式，方便进行搜索与分析。详细请参考**设置云端结构化解析日志**和**设置LTS日志索引配置**。
- 结构化完成后，使用云日志服务LTS提供的**搜索语法**用于设置搜索条件，帮助您更有效地搜索日志。详细请参考**搜索日志**。
- LTS支持使用**SQL分析语法**对结构化后的日志字段进行日志分析，通过统计图表的方式对查询和分析的结果进行可视化展示。详细请参考**分析LTS日志**。

5.2 设置云端结构化解析日志

5.2.1 日志结构化概述

日志数据可分为结构化数据和非结构化数据。结构化数据指能够用数字或统一的数据模型加以描述的数据，具有严格的长度和格式。非结构化数据指不便于用数据库二维逻辑表来表现的数据，数据结构不规则或不完整，没有预定义的数据模型。

日志结构化是以日志流为单位，通过不同的日志提取方式将日志流中的日志进行结构化，提取出有固定格式或者相似程度较高的日志，过滤掉不相关的日志，以便对结构化后的日志按照SQL语法进行查询与分析。

注意事项

- 日志结构化是以日志流为单位，请先创建一个日志流。
- 日志流中的大部分日志需有一定的规则，否则结构化是无意义的。
- 结构化配置修改后，对新写入的日志数据生效，历史日志数据不会生效。

创建结构化配置

通过对日志流添加提取规则将日志流中的原始日志按一定的规律进行提取，并将提取后的日志整合到一起，以便进行SQL查询与分析。

下面详细介绍原始日志结构化的操作步骤：

步骤1 登录LTS控制台，在左侧导航栏中选择“日志管理”。

步骤2 结构化日志以日志流为单位，请在“日志管理”页面选择目标日志组和日志流。

步骤3 在日志流详情页面，单击右上角  ，在弹出页面中，选择“云端结构化解析”，进入日志结构化配置页面，选择对应的日志提取方法进行配置。

- **正则分析**
- **JSON**
- **分隔符**
- **Nginx**
- **结构化模板**

结构化后的日志数据可理解为数据库中的二维表，接下来就可以使用SQL语句对提取的字段进行查询与分析。

说明

- 开启“自动配置索引和快速分析”开关，将使用结构化字段配置字段索引，同时会打开快速分析按钮。开启该按钮后，默认将内置字段hostIP、hostName和pathFile配置为索引字段，并打开快速分析。
- 如果结构化后的字段长度超过20k字节时，仅会保留前20k字节长度。
- 结构化不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime、category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。

步骤4 开启**自定义日志时间**。

步骤5 完成后，单击“保存”。

----结束

修改结构化配置

结构化配置创建完成后，如果您需要修改结构化配置时，操作步骤如下：

步骤1 在结构化配置页面中，单击  ，可修改结构化配置。

说明

- 修改结构化配置支持修改结构化方式、日志提取字段和tag字段等。
- 系统模板不支持修改。

步骤2 完成后，单击“保存”。

----结束

删除结构化配置

如果日志结构化配置不再使用，可以删除结构化配置，操作步骤如下：

步骤1 在结构化配置页面中，单击 ，可删除结构化配置。

步骤2 在弹出对话框中，单击“确定”。

说明

删除结构化配置后，无法恢复，请谨慎操作。

----结束

5.2.2 设置日志云端结构化解析

云日志服务（LTS）目前支持5种日志结构化方式，分别是正则分析、JSON、分隔符、Nginx和结构化模板。您可以根据日志内容的实际场景进行选择。

使用限制

结构化字段最大长度为16KB，超过部分会被截断。

正则分析

正则分析是使用正则表达式提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。

- **从已有日志中选择**：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
- **从剪切板中粘贴**：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

  说明

  时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

  - 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
  - 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
  - 自定义：表示查询指定时间范围的日志数据

步骤2 字段提取。包括自动生成和手动输入两种方式，可将选择的日志提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

- 自动生成：当用户选择自动生成时，可以用鼠标选中示例日志中待结构化的日志内容，在弹出的对话框中为选中内容设置一个名称，名称必须以字母开始，且仅包含字母和数字，单击“添加”。
- 手动输入：当用户选择手动输入时，可以在输入框中输入正则表达式，单击“生成字段”来进行字段提取。正则表达式通过分组来捕获字段，分组指用圆括号"()"括起来的正则表达式，匹配出的内容就表示一个分组，分组包含如下三种形式：

- (exp)：把括号内的正则作为一个分组，系统自动分配组号，规则为从正则表达式的左边开始，第一个左括号”(”对应第一个分组，第二个”(”对应第二个分组，依次类推，组号从1开始，从左向右，依次累加。
- (?<name>exp)：表示命名分组，分组的正则表达式为exp，分组名为name。分组名必须以字母开始，且仅包含字母和数字，可以通过分组名或分组号引用该分组。
- (?:exp)：表示不捕获分组，该分组只在当前位置匹配文本，在该分组之后，无法引用该分组，因为该分组没有分组名，没有分组号，也不会占用分组编号。

说明

- 分词符指将日志内容切分为多个单词的符号，默认分词符包括, '";=()[]{}@&<>/:\\?\n\t\r，在日志搜索或者对日志进行结构化时，可以选取相邻两分词符之间的单词。
- 在手工输入方式中，正则表达式的长度不能超过5000个字符，不强制要求用户在输入正则表达式时对分组进行命名，单击“生成字段”会以命名分组中的分组名作为字段名称，对于非命名分组会提取出对应的字段，并给字段名称默认命名field1、field2、field3……。

步骤3 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

JSON

JSON是通过提取JSON字段将其拆分为键值对。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤2 字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

将以下原始日志输入待操作框中。

```
{"a1": "a1", "b1": "b1", "c1": "c1", "d1": "d1"}
```


说明

- 当日志提取字段的类型为float时，精度为16位有效数字。如果超过16位有效数字，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。
- 当日志提取字段的类型为long时，日志内容超过16位有效数字，只会精确显示前16位有效数字，后面的数字会变为0。
- 当日志提取字段的类型为long时，日志内容超过21位有效数字，则会识别为float类型，建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤3 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

分隔符

分隔符是使用分隔符（例如：逗号、空格或字符）提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 在“步骤2 指定分隔符”需要根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。

说明

- 不可见字符需要输入0x开头的16进制字符，长度为0-4个字符，总共32个不可见字符。
- 自定义字符支持输入1-10个字符，每个字符都作为独立的分隔符。
- 自定义字符串支持输入1-30个字符，字符串整体作为一个分隔符。

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤3字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

将以下原始日志输入待操作框中。

```
1 5f67944957444bd6bb4fe3b367de8f3d 1d515d18-1b36-47dc-a983-bd6512aed4bd 192.168.0.154 192.168.3.25 38929 53 17 1 96 1548752136 1548752736 ACCEPT OK
```


说明

当日志提取字段的类型为float时，精确度为7位有效数字。

如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

Nginx

Nginx是通过log_format指令来自定义访问日志的格式。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 在“步骤2 输入Nginx日志配置”中需要输入Nginx日志配置，根据输入或选择的日志进行配置。其中有默认配置可使用，单击“默认Nginx配置”即可。

说明

标准Nginx配置文件中，日志配置的部分通常以log_format开头。

日志格式

- 默认配置如下所示。

```
log_format  main   '$remote_addr - $remote_user [$time_local] "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';
```

- 用户也可进行自定义配置，具体配置格式要求如下所示。
  - 使用Nginx配置，不可为空
  - 以log_format开头，并且包含（'）和字段名称
  - 长度最大限制为5000
  - 需要与示例日志内容匹配
  - log_format字段之间的间隔，除大小字母、数字、下划线及中划线外，可使用其他任意字符
  - 以（'）或者（';）结尾

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤3 字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

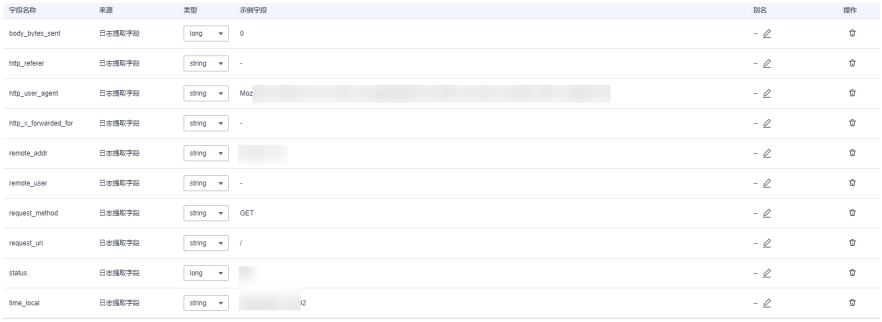
将以下原始日志输入待操作框中。

```
39.149.31.187 - - [12/Mar/2020:12:24:02 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36" "-"
```

并使用如下Nginx日志配置。

```
log_format  main   '$remote_addr - $remote_user [$time_local] "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';
```

图 5-1 智能提取结果



说明

- 当日志提取字段的类型为float时，精确度为7位有效数字。
- 如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

结构化模板

结构化模板是通过自定义模板或系统内置模板提取字段。

详情请参考**结构化模板**。

5.2.3 设置云端结构化字段

使用限制

结构化字段最大长度为16KB，超过部分会被截断。

设置结构化字段

在进行结构化配置字段提取之后，可对结构化字段进行设置，具体设置规则如下表。

表 5-1 结构化字段设置规则

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除
正则分析（自动生成）	用户自定义。 名称必须以字母开始，且仅包含字母和数字。	是	是
正则分析（手动输入）	• 支持在输入正则表达式时进行命名。 • 支持使用系统默认命名field1、field2、field3……，或对其修改后的名称。	是	是
JSON格式	智能提取字段名称，可定义别名。	是	是
分隔符	默认名称field1、field2、field3……，可进行修改。	是	是
Nginx	根据Nginx配置生成，可定义别名。	是	是
自定义模板	用户自定义。	是	是

说明

正则分析（手动输入）、JSON格式、分隔符、Nginx和自定义模板的字段名称需要满足如下要求：

- 只支持输入英文、数字、中划线、下划线及小数点。
- 不能以小数点、下划线开头或以小数点结尾。
- 长度为1-64个字符。

设置 tag 字段

设置结构化配置时，可以对日志维度信息进行tag字段设置，设置完成后可以在可视化界面对设置字段进行SQL查询。

步骤1 在字段提取步骤中选择“tag字段”页签。

步骤2 单击“添加字段”。

步骤3 在tag字段列表中“字段名称”，输入需要设置 tag字段名称，例如hostIP。

说明

tag字段功能上线前设置的结构化配置，在修改结构化配置进行tag字段设置时，系统tag不会带出示例字段。

步骤4 如需添加多个字段可单击“添加字段”，继续添加。

步骤5 设置完成后单击“保存”。

说明

- tag支持的系统字段包括：category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。
- tag不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime。
- 日志提取字段和tag字段可以同时设置。

----结束

5.2.4 设置云端结构化自定义日志时间

当日志接入云日志服务（LTS）时，您可以通过开启“自定义日志时间”开关，将日志中的时间字段设置为接入配置的时间。

开启自定义日志时间

步骤1 在左侧导航栏中选择“日志管理”。

步骤2 结构化日志以日志流为单位，在“日志管理”页面选择目标日志组和日志流。

步骤3 单击日志流名称进入日志流详情页面，单击右上角  ，在弹出页面中，选择“云端结构化解析”，进入日志结构化配置页面，选择对应的日志提取方法进行配置。

步骤4 配置完成后，开启自定义日志时间开关  ，配置如下参数。

表 5-2 参数配置表

参数	说明	示例
字段key	已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。	test
字段value	已提取的字段value，选择字段key后，将自动填充。	2022-07-19 12:12:00
时间格式	请参考**常见日志时间格式**。	yyyy-MM-dd HH:mm:ss
操作	单击“校验”，提示“时间格式和字段value匹配成功”则表示校验成功。	-

说明

切换自定义日志时间开关时，可能会导致日志搜索界面在切换时间点附近出现时间偏差，请勿频繁切换自定义日志时间开关。

----结束

常见日志时间格式

支持的常见日志时间格式如下表所示。

说明

默认情况下，日志服务中的日志时间戳精确到秒，所以时间格式只需配置到秒，无需配置毫秒、微秒等信息。

表 5-3 时间格式

时间格式	说明	示例
EEE	星期的缩写。	Fri
EEEE	星期的全称。	Friday
MMM	月份的缩写。	Jan
MMMM	月份的全称。	January
dd	每月第几天，十进制，范围为01~31。	07, 31
HH	小时，24小时制。	22
hh	小时，12小时制。	11
MM	月份，十进制，范围为01~12。	08
mm	分钟，十进制，范围为00~59。	59
a	AM或PM。	AM、PM
hh:mm:ss a	12小时制的时间组合。	11:59:59 AM
HH:mm	小时和分钟组合。	23:59
ss	秒数，十进制，范围为00~59。	59
yy	年份，十进制，不带世纪，范围为00~99。	04、98
yyyy	年份，十进制。	2004、1998
d	每月第几天，十进制，范围为1~31。 如果是个位数字，前面需要加空格。	7、31
DDD	一年中的天数，十进制，范围为001~366。	365
u	星期几，十进制，范围为1~7，1表示周一。	2
w	每年的第几周，星期天是一周的开始，范围为00~53。	23

时间格式	说明	示例
w	每年的第几周，星期一是一周的开始，范围为01~53。 如果一月份刚开始的一周>=4天，则认为是第1周，否则认为下一个星期是第1周。	24
U	星期几，十进制，范围为0~6，0代表周日。	5
EEE MMM dd HH:mm:ss yyyy	标准的日期和时间。	Tue Nov 20 14:12:58 2020
EEE MMM dd yyyy	标准的日期，不带时间。	Tue Nov 20 2020
HH:mm:ss	标准的时间，不带日期。	11:59:59
%s	Unix时间戳。	147618725

示例

常见的时间标准、示例及对应的时间表达式如下所示。

表 5-4 示例

示例	时间表达式	时间标准
2022-07-14T19:57:36+08:00	yyyy-MM-dd'T'HH:mm:ssXXX	自定义
1548752136	%s	自定义
27/Jan/2022:15:56:44	dd/MMM/yyyy:HH:mm:ss	自定义
2022-08-15 17:53:23+08	yyyy-MM-dd HH:mm:ssX	自定义
2022-08-05T08:24:15.536+0000	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-20T10:04:03.204000Z	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-22T06:52:08Z	yyyy-MM-dd'T'HH:mm:ssZ	自定义
2022-07-24T10:06:41.000	yyyy-MM-dd'T'HH:mm:ss.SSS	自定义
[2022-12-11 15:05:07.012]	[yyyy-MM-dd HH:mm:ss.SSS]	自定义
Monday, 02-Jan-06 15:04:05 MST	EEEE, dd-MMM-yy HH:mm:ss Z	RFC850

示例	时间表达式	时间标准
Mon, 02 Jan 2006 15:04:05 MST	EEE, dd MMM-yyyy HH:mm:ss Z	RFC1123
02 Jan 06 15:04 MST	dd MMM yy HH:mm Z	RFC822
02 Jan 06 15:04 -0700	dd MMM yy HH:mm Z	RFC822Z
2023-01-02T15:04:05.999999999Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339Nano
2023-01-02T15:04:05Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339
2022-12-11 15:05:07	yyyy-MM-dd HH:mm:ss	自定义

5.2.5 设置云端结构化模板

云日志服务（LTS）目前支持的结构化模板有两种：系统模板和自定义模板。

系统模板

支持多种系统模板，不支持修改系统模板的字段类型和删除字段，详情请参考表5-5。

步骤1 在“选择模板”下，选择“系统模板”，选择对应的系统模板，模板日志从对应的云服务接入，可以直接应用模板的数据模型作为示例日志。

步骤2 选择模板后“模板详情”中会自动显示对应的日志解析结果。单击“保存”完成结构化配置。

说明

- 结构化配置时，如果使用系统模板，则系统模板中的时间为自定义日志时间。支持通过模板名称搜索模板，方便用户快速查询模板信息。
- string类型的字段不支持使用运算符（>=<）或 in 语法进行范围查询，建议使用星号（*）或问号（?）进行模糊查询。需要重新配置结构化，将该字段修改为数字类型。

表 5-5 系统模板

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
ELB模板	根据ELB资料中提供的日志字段被定义。	否	否	弹性负载均衡 ELB接入LTS
VPC模板	根据VPC资料中提供的日志字段被定义。	否	否	虚拟私有云VPC接入LTS
CTS模板	字段名称为json日志中的key。	否	否	云审计服务CTS接入LTS

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
APIG模板	根据APIG资料中提供的日志字段被定义。	否	否	API网关APIG接入LTS
DCS审计日志	根据DCS资料中提供的日志字段被定义。	否	否	分布式缓存服务DCS接入LTS
TOMCAT	根据TOMCAT官网提供的字段名称进行nginx解析的名称。	否	否	TOMCAT结构化模板字段介绍
NGINX	根据NGINX资料中提供的日志字段被定义。	否	否	NGINX结构化模板字段介绍
GAUSSV5审计日志	根据GAUSSV5资料中提供的日志字段被定义。	否	否	云数据库GaussDB接入LTS
DDS审计日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
DDS错误日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
DDS慢日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
CFW访问控制日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
CFW攻击日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
CFW流量日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
MYSQL错误日志	根据MYSQL资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
MYSQL慢日志	根据MYSQL资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS
POSTGRESQL慢日志	根据POSTGRESQL慢日志资料中提供的日志字段被定义。	否	否	云数据库RDS for PostgreSQL接入LTS
POSTGRESQL错误日志	根据POSTGRESQL错误日志资料中提供的日志字段被定义。	否	否	云数据库RDS for PostgreSQL接入LTS
SQLSERVER错误日志	根据SQLSERVER资料中提供的日志字段被定义。	否	否	云数据库RDS for SQLServer接入LTS
GeminiDB Redis慢日志	根据GeminiDB Redis资料中提供的日志字段被定义。	否	否	云数据库GeminiDB接入LTS
CDN	根据CDN资料中提供的日志字段被定义。	否	否	内容分发网络CDN接入LTS
SMN	根据SMN资料中提供的日志字段被定义。	否	否	消息通知服务SMN接入LTS
GAUSSDB_MYSQL错误日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	云数据库GaussDB(for MySQL)接入LTS
GAUSSDB_MYSQL慢日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	云数据库GaussDB(for MySQL)接入LTS
ER企业路由器	根据ER企业路由器资料中提供的日志字段被定义。	否	否	企业路由器ER接入LTS

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
MYSQL审计日志	根据MYSQL审计日志资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS
GeminiDB Cassandra慢日志	根据GeminiDB Cassandra慢日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Cassandra接入LTS
GeminiDB Mongo慢日志	根据GeminiDB Mongo慢日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Mongo接入LTS
GeminiDB Mongo错误日志	根据GeminiDB Mongo错误日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Mongo接入LTS
WAF访问日志	根据WAF访问日志资料中提供的日志字段被定义。	否	否	Web应用防火墙WAF接入LTS
WAF攻击日志	根据WAF攻击日志资料中提供的日志字段被定义。	否	否	Web应用防火墙WAF接入LTS
DMS重平衡日志	根据DMS重平衡日志资料中提供的日志字段被定义。	否	否	分布式消息服务Kafka版接入LTS
CCE审计日志	根据CCE审计日志资料中提供的日志字段被定义。	否	否	云容器引擎CCE应用日志接入LTS
CCE事件日志	根据CCE事件日志资料中提供的日志字段被定义。	否	否	云容器引擎CCE应用日志接入LTS
GeminiDB Redis审计日志	根据GeminiDB Redis审计日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB接入LTS
influx慢日志	根据influx慢日志资料中提供的日志字段被定义。	否	否	-

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
METRIC系统日志	根据日志生成指标任务过滤的监控日志字段被定义。	否	否	-
Microgateway	根据Microgateway应用网关提供的日志字段被定义。	否	否	-
GeminiDB Mongo接口审计日志	根据GeminiDB Mongo接口审计日志提供的日志字段被定义。	否	否	-

----结束

自定义模板

在“选择模板”下，选择“自定义模板”，选择已有的结构化模板。模板来源有以下两种方式：

- 在配置正则分析、JSON、分隔符或Nginx方式时单击左下角的“另存为模板”，系统会弹出“另存模板”页面，输入模板名称，单击“确定”，完成自定义模板的保存，会在“自定义模板”下的模板列表看到该模板。
- 新增结构化模板，具体操作如下：

  在“选择模板”下，选择“自定义模板”，单击“新增结构化模板”，在“新增结构化模板”界面选择正则分析、JSON、分隔符或Nginx方式，进行配置，配置完成后输入模板名称，单击“确定”。完成自定义模板的保存，会在“自定义模板”下的模板列表看到该模板。

5.2.6 结构化系统模板字段详情

5.2.6.1 NGINX 结构化模板字段介绍

介绍NGINX接入日志结构化模板详情。

结构化模板日志详情

- NGINX日志示例

  表 5-6 结构化模板示例

模板名称	示例日志
NGINX	192.168.1.101 - [27/Aug/2018:14:20:29 +0800] "GET http://www.example.com / HTTP/1.0" 200 8796 6775 "-" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.131 Safari/537.36" "-" "0.185" "0.010" 12.129.120.121:8090 200 794

- 结构化字段及字段说明

表 5-7 结构化字段

字段	示例	描述	类型
body_bytes_sent	6775	发送给客户端的文件主体内容的大小	long
bytes_sent	8796	响应大小	long
host	www.example.com	请求域名	string
http_referer	-	来源页面链接	string
http_user_agent	Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.131 Safari/537.36	客户端浏览器信息	string
http_x_forwarded_for	-	客户端的真实ip	string
remote_addr	192.168.1.101	客户端地址	string
remote_user	-	远程客户端用户名称	string
request_length	794	请求长度	long
request_method	GET	请求方法	string
request_time	0.185	请求时间	float
request_uri	/	请求url	string
scheme	http	http或https	string
server_protocol	HTTP/1.0	请求协议	string
status	200	响应状态码	long
time_local	27/Aug/2018:14:20:29	日期	string
upstream_addr	12.129.120.121:8090	后端服务器地址	string
upstream_response_time	0.010	后端服务器响应时间	float
upstream_status	200	后端服务器响应状态	long

5.2.6.2 TOMCAT 结构化模板字段介绍

介绍TOMCAT接入日志结构化模板详情。

结构化模板日志详情

- TOMCAT日志示例

  表 5-8 结构化模板示例

模板名称	示例日志
TOMCAT	192.168.12.2 - - [07/Mar/2018:09:49:55 +0800] "GET /logHello/test HTTP/1.1" 200 1943

- 结构化字段及字段说明

  表 5-9 结构化字段

字段	示例	描述	类型
bytes_sent	1943	发送的字节	long
remote_ip_address	192.168.12.2	远程IP地址	string
remote_logical_username	-	远程逻辑从identd的用户名	string
remote_user_authenticated	-	远程用户身份验证	string
router_uri	/logHello/test	请求url	string
scheme	GET	请求方法	string
server_protocol	HTTP/1.1	请求协议	string
status	200	响应的http状态码	long
time_local	07/Mar/2018:09:49:55	日期	string

5.3 设置 LTS 日志索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

日志示例

以下是一条典型日志，content字段值是日志原文，使用分隔符逗号将原始日志解析成3个字段level、status、message；

示例日志中的hostName、hostIP、pathFile是常见的内置保留字段，详细内置字段请参考**内置保留字段**。

```
{
"hostName":"epstest-xx518",
"hostIP":"192.168.0.31",
"pathFile":"stdout.log",
"content":"error,400,I Know XX",
"level":"error",
"status":400,
"message":"I Know XX"
}
```

索引类型

云日志服务的索引类型如下：

表 5-10 索引类型表

索引类型	说明
全文索引	开启全文索引后，日志服务根据您设置的分词符将整条日志所有字段值拆分成多个词并构建索引。 **说明** • 用户上传的自定义标签（label）字段，不包含在全文索引中，如果您需要搜索自定义标签字段，请添加对应的字段索引。 • LTS内置保留字段，不包含在全文索引中，您需要通过字段索引Key:Value的方式进行搜索，请参考**内置保留字段**。
字段索引	配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询，缩小查询范围。 **说明** • 日志服务默认为部分内置保留字段创建字段索引，请参考**内置保留字段**。 • 如果您的某个字段单独配置了字段索引，那么该字段值的分词符以字段索引配置为准。 • 结构化配置中的的快速分析列已被移除，如果您要使用**快速分析功能**，则必须配置字段索引且开启对应字段的快速分析按钮。 关于日志示例有两种情况： • 在日志示例中，配置了level和status两个字段索引，其中level是string类型，字段值是error，单独配置了分词符，status是long类型，不需要配置分词符；您可以使用level : error的方式精确搜索level字段值为error的所有日志。 • 在日志示例中，云日志服务LTS会默认为hostName、hostIP、pathFile这些内置保留字段创建字段索引。

注意事项

- 全文索引属性和字段索引属性必须至少启用一种。
- 创建索引会产生索引流量和索引存储空间，费用说明请参见**价格计算器**。
- 索引配置（新增、编辑、删除字段，修改配置项等操作）只对新写入的日志生效，历史日志不会生效。当前不支持对历史日志重建索引。
- 关闭索引后，历史索引的存储空间将在当前日志流的数据保存时间到期后，自动被清除。
- 日志服务默认已为部分内置保留字段创建字段索引，请参见**内置保留字段**。

- 不同的索引配置，会产生不同的查询和分析结果，请根据您的需求，合理创建索引。全文索引和字段索引互不影响。
- 索引配置修改后，对新写入的日志数据生效，历史日志数据不会生效。
- 在字段索引功能上线前，SQL分析支持的字段来自于云端结构化解析；在字段索引功能上线后，只要用户配置了字段索引，SQL分析支持的字段将来自于字段索引，因此修改字段索引可能对现有的可视化图表、仪表盘、SQL告警、定时SQL、Grafana接入中的查询结果产生影响，请谨慎操作！
- 字段索引配置为JSON数据类型时，在原始日志页面不支持对JSON子字段使用快速分析、跳转图表，不支持可视化；在搜索分析页面下支持JSON子字段使用快速分析和SQL可视化功能。
- 字段索引配置为JSON数据类型时，对JSON父字段查询时支持对命中结果高亮，对JSON子字段查询时不支持对命中结果高亮。

索引流量

创建索引后，会产生索引流量，索引流量的详细计算规则请参考**价格计算器**。

内置保留字段

在采集日志时，日志服务会将采集时间、日志类型、主机IP等信息以Key-Value对的形式添加到日志中，这些字段是云日志服务的内置字段。

说明

- 使用API写入日志数据或添加ICAgent配置时，请不要将字段名称设置为内置保留字段，否则可能会造成字段名称重复、查询不精确等问题。
- 日志服务为日志数据增加的内置保留字段当前免费，后续会按照按量付费方式正常收费（为其开启索引时也会产生少量索引流量及存储费用）。更多信息请参见**价格计算器**。
- 用户自定义日志字段名称中不能使用双下划线__，否则无法配置索引。

表 5-11 内置保留字段说明

内置保留字段	数据格式	索引与统计设置	说明
collectTime	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为collectTime创建字段索引，索引数据类型为long类型。 查询时输入collectTime : xxx。	采集时间，指日志被采集器ICAgent采集时的时间。 示例中的"collectTime":"1681896081334"，转换成标准时间是2023-04-19 17:21:21

内置保留字段	数据格式	索引与统计设置	说明
__time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为time创建字段索引，索引数据类型为long类型。该字段不支持查询。	日志时间，指的是日志在控制台页面展示的日志时间。 例如示例中的"__time__":"1681896081334"，转换成标准时间是2023-04-19 17:21:21 日志时间默认使用采集时间，也支持自定义日志时间。
lineNum	整型	索引设置：开启索引后，日志服务默认为lineNum创建字段索引，索引数据类型为long类型。	行号（偏移量），用来排序日志。 非高精度日志会根据collectTime生成，默认是collectTime * 1000000 + 1，高精度日志就是用户上报的纳秒值。 例如示例中的"lineNum":"1681896081333991900"。
category	字符串	索引设置：开启索引后，日志服务默认为category创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入category: xxx。	日志类型，表示该日志的来源。 例如ICAgent采集的日志该字段为LTS，某云服务例如DCS上报的日志该字段为DCS。
clusterName	字符串	索引设置：开启索引后，日志服务默认为clusterName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterName: xxx。	集群名称，k8s场景下集群名称。 例如示例中的"clusterName":"epstest"。
clusterId	字符串	索引设置：开启索引后，日志服务默认为clusterId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterId: xxx。	集群ID，k8s场景下集群ID。 例如示例中的"clusterId":"c7f3f4a5-xxxx-11ed-a4ec-0255ac100b07"。

内置保留字段	数据格式	索引与统计设置	说明
nameSpace	字符串	索引设置：开启索引后，日志服务默认为nameSpace创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入nameSpace: xxx。	命名空间，k8s场景下命名空间。 例如示例中的"nameSpace":"monitoring"。
appName	字符串	索引设置：开启索引后，日志服务默认为appName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入appName: xxx。	组件名称，k8s场景下工作负载的名称。 例如示例中的"appName":"alertmanager-alertmanager"。
serviceID	字符串	索引设置：开启索引后，日志服务默认为serviceID创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入serviceID: xxx。	工作负载ID，k8s场景下工作负载ID。 例如示例中的"serviceID":"cf5b453xxxad61d4c483b50da3fad5ad"。
podName	字符串	索引设置：开启索引后，日志服务默认为podName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podName: xxx。	POD名称，k8s场景下POD名称。 例如示例中的"podName":"alertmanager-alertmanager-0"。
podIp	字符串	索引设置：开启索引后，日志服务默认为podIp创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podIp: xxx。	pod的ip，k8s场景下pod的IP地址。 例如示例中的"podIp":"10.0.0.145"。
containerName	字符串	索引设置：开启索引后，日志服务默认为containerName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入containerName: xxx。	容器名称，k8s场景下容器名称。 例如示例中的"containerName":"config-reloader"。

内置保留字段	数据格式	索引与统计设置	说明
hostName	字符串	索引设置：开启索引后，日志服务默认为hostName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostName: xxx。	主机名称，ICAgent所在主机的名称。 例如示例中的"hostName":"epstest-xx518"。
hostId	字符串	索引设置：开启索引后，日志服务默认为hostId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostId: xxx。	主机ID，ICAgent所在主机的id，该id由ICAgent生成。 例如示例中的"hostId":"318c02fe-xxxx-4c91-b5bb-6923513b6c34"。
hostIP	字符串	索引设置：开启索引后，日志服务默认为hostIP创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIP: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv4场景） 例如示例中的"hostIP":"192.168.0.31"。
hostIPv6	字符串	索引设置：开启索引后，日志服务默认为hostIPv6创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIPv6: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv6场景） 例如示例中的"hostIPv6":""。
pathFile	字符串	索引设置：开启索引后，日志服务默认为pathFile创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入pathFile: xxx。	文件路径，采集的日志文件的路径。 例如示例中的"pathFile":"stdout.log"。
content	字符串	索引设置：开启全文索引后，会使用全文索引定义的分词符对content字段的value进行分词；不支持将content字段配置到字段索引中。	日志原文 例如示例中的"content":"level=error ts=2023-04-19T09:21:21.333895559Z"

内置保留字段	数据格式	索引与统计设置	说明
__receive_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__receive_time__创建字段索引，索引数据类型为long类型。	上报日志的服务端时间，相当于LTS采集端接收到日志的时间。
__client_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__client_time__创建字段索引，索引数据类型为long类型。	端侧日志的客户端上报时间。
_content_parse_fail_	字符串	索引设置：开启索引后，日志服务默认为_content_parse_fail_创建字段索引，索引数据类型为string类型，分词字符为默认分词符。查询时输入_content_parse_fail_:xxx。	上报日志解析失败的日志内容。
__save_time__	整型，Unix时间戳（毫秒）	不支持将__save_time__字段配置到字段索引中。	日志流引擎的时间字段，根据此时间拉取对应时间段内的日志数据。
__time	整型，Unix时间戳（毫秒）	不支持将__time字段配置到字段索引中。	不涉及。
logContent	字符串	不支持将logContent字段配置到字段索引中。	不涉及。
logContentSize	整型	不支持将logContentSize字段配置到字段索引中。	不涉及。
logIndexSize	整型	不支持将logIndexSize字段配置到字段索引中。	不涉及。
groupName	字符串	不支持将groupName字段配置到字段索引中。	不涉及。
logStream	字符串	不支持将logStream字段配置到字段索引中。	不涉及。

配置全文索引

步骤1 登录云服务日志控制台，单击“日志管理”。

步骤2 在日志组列表中，单击日志组名称左侧的 ，选择日志流，进入日志流管理界面。

步骤3 在日志流详情页面，单击右上角 ，进入索引配置页面。

步骤4 在索引配置页面中，默认开启“全文索引”按钮。

说明

- 在索引配置页面选择自动配置时，默认获取最近15分钟的原始日志和内置字段的交集，LTS自动将原始日志和内置字段的交集、当前结构化字段、tag字段一起组成字段索引下方的表格数据。
- 若15分钟内没有原始日志，则获取hostIP、hostName、pathFile、结构化字段、tag字段结合共同组成字段索引下方的表格数据。
- ECS接入选择结构化配置时，进入索引配置页面，则会自动加上如下字段：category、hostName、hostId、 hostIP、 hostIPv6、 pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。
- CCE接入选择结构化配置时，进入索引配置页面，则会自动加上如下字段：category、clusterId、 clusterName、 nameSpace、 podName、 containerName、 appName、hostName、 hostId、 hostIP、 hostIPv6、 pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。

步骤5 请参考表5-12配置参数信息。

表 5-12 自定义全文索引配置参数

参数	说明
全文索引	打开全文索引开关，表示创建全文索引。
大小写敏感	查询时是否区分英文字母的大小写。 • 打开大小写敏感开关，则查询时区分大小写。例如示例日志含有Know，那么您只能使用**Know**才能查询到该日志。 • 关闭大小写敏感开关，则查询时不区分大小写。例如示例日志含有Know，那么您使用关键字**KNOW**和**know**都能查到该日志。

参数	说明
包含中文	查询时是否区分中英文。 ● 打开包含中文开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 **说明** – 一元分词是指将中文字符串拆分为一个个独立的中文字。 – 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 – 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 ● 关闭包含中文开关后，按照分词符的设置拆分所有内容。 例如示例日志内容为： **error,400,I Know 今天是星期一**。 ● 关闭包含中文开关后，按照分词符的设置拆分英文内容，日志会被拆分为**error**、**400**、**I**、**Know**、**今天是星期一**，您可以通过**error**或**今天是星期一**查找该日志。 ● 打开包含中文开关后，日志服务后台分词器将日志拆分为**error**、**400**、**I**、**Know**、**今**、**天**、**是**、**星**、**期**、**一**，您通过**error**或**今天**等词都可以查找到该日志。
分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。所有的ASCII码包括中文都可被定义为分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 例如示例日志内容为： **error,400,I Know 今天是星期一**。 ● 如果不设置任何分词符，整条日志被作为一个词**error,400,I Know 今天是星期一**，您只能通过完整字符串**error,400,I Know 今天是星期一**或模糊查询**error,400,I K***查找该日志。 ● 如果设置分词符为逗号（,），则原始日志被拆分为**error**、**400**、**I Know 今天是星期一**3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**error**、**400**、**Kn***、**今天是***。 ● 如果设置分词符为逗号（,）和空格，则原始日志被拆分为**error**、**400**、**I**、**Know**、**今天是星期一**5个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**Know**、**今天是***。
特殊分词符	单击“添加特殊分词符”，参考**ASCII码对照表**输入ASCII值。

步骤6 完成后，单击确定。

----结束

配置字段索引

创建字段索引时，最多支持添加500个字段。其中JSON类型字段，最多支持添加100个子字段。

说明

字段索引的自定义分词符和特殊分词符仅支持白名单用户提交工单申请使用。详细操作请参考**提交工单**。

步骤1 配置全文索引后，在索引配置页面的日志分析下方，单击开启可视化后，配置的字段索引支持SQL可视化分析，否则无法查询到ICAgent结构化的可视化数据。

步骤2 在索引配置页面的字段索引下方，单击“添加字段”，参考**表5-13**设置字段信息。

步骤3 或者勾选字段，单击批量配置，在批量配置页面设置参数。

图 5-2 批量配置

步骤4 参考**表5-13**配置字段索引。

说明

- 字段索引的参数配置仅对该字段生效。
- 当添加的字段在日志内容中不存在时，则配置的该索引字段无效。

表 5-13 自定义字段索引配置参数

参数	说明
字段名称	日志字段名称，例如示例日志中的level。 字段名称只能包括字母、数字或下划线（_），且只能以字母或下划线（_）开头，字段名称中不能含有双下划线。 **说明** • 双下划线（__）在LTS不对用户呈现的内置保留字段中使用，用户自定义日志字段名中不能使用双下划线__，否则无法配置字段索引名称。 • 日志服务默认会对部分内置保留字段开启字段索引，请参见**内置保留字段**。 • 若是内置字段，在字段名称后会显示“内置”字眼，方便用户识别。
执行操作	显示字段的添加状态：新增、不修改、修改、删除。索引字段有变动后，单击“修改对比”，即可查看原配置内容与修改后配置内容的差异。 • 显示新增的字段不支持修改执行操作。 • 修改类型、大小写敏感、自定义分词符、特殊分词符、包含中文、快速分析时，会与原索引配置中的字段进行对比，若任意一项不同，则执行操作变为“修改”。 • 索引配置单击确定后，不会保存执行操作为“删除”的字段。
类型	• 日志字段值（Value）的数据类型，可选值为string、long、float、json。 **说明** 字段json类型只对ICAgent结构化解析生效，对云端结构化解析不生效。 • long类型和float类型不支持设置大小写敏感、包含中文和分词符。
大小写敏感	查询时是否区分英文字母的大小写。 • 打开大小写敏感开关，则查询时区分大小写。例如示例日志message字段中含有Know，那么您只能使用**message:Know**才能查询到该日志。 • 关闭大小写敏感开关，则查询时不区分大小写。例如示例日志message字段中含有Know，那么您使用关键字**message:KNOW**和**message:know**都能查到该日志。

参数	说明
自定义分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。所有的ASCII码包括中文都可被定义为分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 例如示例日志message字段内容为：**I Know 今天是星期一**。 • 如果不设置任何分词符，整条日志被作为一个词**I Know 今天是星期一**，您只能通过完整字符串**message:I Know 今天是星期一**或模糊查询**message:I Know 今天是*查找该日志**。 • 如果设置分词符为空格，则原始日志被拆分为**I**、**Know**、**今天是星期一**3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**message:Know**或**message:今天是星期一**。
特殊分词符	单击“添加特殊分词符”，参考**ASCII码对照表**输入ASCII值。
包含中文	查询时是否区分中英文。 • 打开包含中文开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 **说明** – 一元分词是指将中文字符串拆分为一个个独立的中文字。 – 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 – 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 • 关闭包含中文开关后，按照分词符的设置拆分所有内容。 例如示例日志message字段内容为：**I Know 今天是星期一**。 • 关闭包含中文开关后，按照分词符的设置拆分英文内容，日志会被拆分为**I**、**Know**、**今天是星期一**，您可以通过**message:Know**或**message:今天是星期一**查找该日志。 • 打开包含中文开关后，日志服务后台分词器将日志拆分为**I**、**Know**、**今**、**天**、**是**、**星**、**期**、**一**，您通过**message:Know**或**message:今天**等词都可以查找到该日志。
快速分析	默认为开启状态，开启后，可以对字段值做采样统计，请参见**11.6.4-快速分析**。 **说明** • 快速分析的原理是对搜索命中的日志采样10万条进行数据统计，不是全量统计。 • 快速分析的字段长度最大为2000字节。 • 快速分析字段展示前100条数据。
操作	单击  ，删除添加的自定义字段。

步骤5 完成后，单击“确定”。

----结束

自动配置字段索引

在创建字段索引时，您可以单击自动配置，日志服务会自动添加一些字段索引，您可以根据自己的需要增加或者删除字段：

- 日志服务会根据采集时预览数据中的第一条内容，自动生成字段索引。
- 日志服务会选取几个最常见的内置保留字段添加到字段索引中（例如hostIP、hostName、pathFile）。

ASCII 码对照表

表 5-14 ASCII 码对照表

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL（空字符）	32	空格	64	@	96	`
1	SOH（标题开始）	33	!	65	A	97	a
2	STX（正文开始）	34	"	66	B	98	b
3	ETX（正文结束）	35	#	67	C	99	c
4	EOT（传输结束）	36	$	68	D	100	d
5	ENQ（询问字符）	37	%	69	E	101	e
6	ACK（确认回应）	38	&	70	F	102	f
7	BEL（响铃）	39	'	71	G	103	g
8	BS（退格）	40	(	72	H	104	h
9	HT（水平定位符号，制表符）	41	)	73	I	105	i
10	LF（换行）	42	*	74	J	106	j
11	VT（垂直定位符号）	43	+	75	K	107	k
12	FF（换页键）	44	,	76	L	108	l

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
13	CR（归位键）	45	-	77	M	109	m
14	SO（取消变换）	46	.	78	N	110	n
15	SI（启用变换）	47	/	79	O	111	o
16	DLE（跳出数据通讯）	48	0	80	P	112	p
17	DC1（设备控制1）	49	1	81	Q	113	q
18	DC2（设备控制2）	50	2	82	R	114	r
19	DC3（设备控制3）	51	3	83	S	115	s
20	DC4（设备控制4）	52	4	84	T	116	t
21	NAK（确认失败回应）	53	5	85	U	117	u
22	SYN（同步用暂停）	54	6	86	V	118	v
23	ETB（区块传输结束）	55	7	87	W	119	w
24	CAN（取消）	56	8	88	X	120	x
25	EM（连接介质中断）	57	9	89	Y	121	y
26	SUB（替换）	58	:	90	Z	122	z
27	ESC（跳出）	59	;	91	[	123	{
28	FS（文件分割符）	60	<	92	\	124	|
29	GS（组群分隔符）	61	=	93	]	125	}
30	RS（记录分隔符）	62	>	94	^	126	~
31	US（单元分隔符）	63	?	95	_	127	DEL（删除）

5.4 搜索日志

5.4.1 进入搜索 LTS 日志页面

您可以通过本操作设置关键字和时间范围进行日志搜索。

1. 在云日志服务管理控制台，单击“日志管理”。
2. 在日志组列表中，单击日志组名称前对应的 按钮。
3. 在日志流列表中，单击日志流名称，进入日志详情页面。
4. 在搜索框上方选择时间范围。

   时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

   说明

   - 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
   - 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
   - 自定义：表示查询指定时间范围的日志数据。支持选择3个月的范围，白名单用户支持选择6个月的范围。如有需要，请**提交工单**申请。
5. 在日志详情页面，有以下搜索方式：
   a. 在页面搜索区域中，鼠标单击搜索框，输入待搜索的关键字，或在弹出的下拉框中选择待搜索的字段和关键词，单击“查询”，开始搜索。

      在原始日志页签下方显示包含搜索关键字的日志。

      说明

      - 内置保留字段有appName、category、clusterId、clusterName、collectTime等，默认简化显示，并且hostIP、hostName、pathFile默认显示在最前面。更多信息请参考**日志搜索与分析概述**。
      - 结构化配置的字段按照key:value显示。
      - 日志搜索框内容过多时支持自动换行并多行显示。
      - 支持固定搜索框高度。

   b. 在日志搜索页面中，展示不同时间段的日志总条数柱状图，柱状图旁边显示日志条数刻度。

      说明

      若使用了内嵌功能，支持设置收起或展开式日志条数统计图。关于内嵌参数请参考**云日志服务地址**。

   c. 在“日志搜索”页签，鼠标悬浮指向**日志内容**中的字段，单击蓝色字体的日志内容，支持以下方式搜索日志：复制、添加到查询、从查询中排除、添加到查询（交互模式）、从查询中排除（交互模式）。

   d. 对已创建快速分析的字段，单击选择字段可直接将其添加到页面搜索框中，进行搜索。

> **说明**
>
> 通过单击字段添加到搜索框中，如果是同一字段，则将直接替换该方式添加的字段，不会进行AND搜索；如果是不同字段，则对不同字段进行AND搜索。

e. 在页面搜索区域，使用键盘的"↑""↓"箭头，选择待搜索的关键字或搜索语法，单击Tab键或Enter键选中后，单击“查询”，开始搜索。

6. 在日志内容下方，单击时间前面的 展开图标，支持以Table和JSON形式展示结构化字段。
   - 在Table页签，支持对该字段进行添加到查询、从查询中排除、字段存在、字段不存在、隐藏的方式搜索日志。更多信息请参考**LTS搜索语法介绍**。
   - 在JSON页签，支持查看或者复制日志内容。
7. 版面设置。
   a. 在下拉框单击编辑版面，进入版面设置页面，版面列表自带默认版面、纯净版面、容器日志默认版面，可以设置字段在版面是否显示。

      云端: 适用于有写权限的用户，版面配置信息保存在云端。

      本地缓存: 适用于只有读权限的用户，版面配置信息缓存在本地浏览器。

   b. 单击 新增自定义版面，设置版面名称和版面字段的可见性。
   c. 设置完成后，单击“确定”，返回下拉框显示新增的自定义版面。
8. 交互式搜索分析，适用于生成简单的分析语句，如果您需要使用更多的函数或者嵌套查询，请手动输入SQL语句。
   a. 单击搜索框前面的 按钮，进入交互式搜索分析页面。

   b. 选择日志搜索的字段和条件，搜索框则会展示对应字段的值。
   c. 支持添加关联关系或添加组。
   d. 设置完成后，支持预览搜索语句。
   e. 设置完成后，单击“确定”，即可在搜索框快速生成分析语句。

日志搜索的常用操作

日志搜索的常用操作有分享日志、刷新等操作，具体参考**表5-15**：

表 5-15 常用操作

操作	说明
创建快速查询	单击 按钮，创建快速查询。

操作	说明
查看仪表盘	单击  按钮，在弹出来的仪表盘页面，可查看已创建的仪表盘。
添加告警	单击  按钮，在弹出的页面，支持**新建告警规则**。
分享日志	单击  复制当前日志搜索页面的链接，用于分享搜索日志。
刷新日志	单击  对日志进行刷新，有两种方式刷新方式：手动刷新和自动刷新。 • 手动刷新：单击“手动刷新”可直接对日志进行刷新。 • 自动刷新：选择自动刷新的间隔时间，将对日志进行自动刷新。间隔时间范围为15秒、30秒、1分钟和5分钟。
复制	单击  复制日志内容。
查看上下文	单击  查看日志上下文。 **说明** 支持选择简洁模式查看日志上下文。支持下载上下文内容。
简化字段详情	单击 查看简化字段详情。
换行/取消换行	单击  按钮，搜索的日志内容将换行显示。若不需要换行， 单击  按钮，取消换行。 **说明** 默认开启换行按钮。
下载日志	单击  按钮，在弹出的下载日志页面中单击“本地下载”和“前往创建转储”。 本地下载：将日志文件直接下载到本地，单次下载支持最大5,000条日志。 在下拉框中选择“.csv”或“.txt”，单击“开始下载日志”，可将日志导出至本地。 **说明** • 选择以CSV格式导出日志后，本地以表格形式保存日志的具体标签信息。 • 选择导出TXT格式日志后，本地会以.txt格式保存日志的日志内容。 前往创建转储：通过OBS转储任务下载日志文件，单次下载支持最大100万条日志。 单击“前往创建转储”，跳转至配置转储页面，详细请参考**日志转储**。

操作	说明
全部折叠/全部展开	单击 设置日志内容展示的行数。若不需要展示日志内容，再单击一次 按钮即可关闭展示的日志内容。 **说明** 默认不折叠。折叠后，默认显示2行，最多支持展示6行。
JSON设置	鼠标悬浮在 按钮上，单击“JSON设置”，在弹出的JSON设置页面中，设置格式化显示。 **说明** 默认开启格式化，JSON默认展开层级为2层。若日志包含多个反斜杠，当日志展示为json格式时，会丢失一个反斜杠，因为json解析会将第一个反斜杠作为转义符处理。 ● 开启格式化按钮：设置JSON默认展开层级，最大设置为10层。 ● 关闭格式化按钮：对于JSON格式的日志，将不会格式化层级显示。
日志折叠设置	鼠标悬浮在 按钮上，单击“日志折叠设置”，在弹出的日志折叠设置页面中，设置长日志字符个数。 日志超过设置的长日志字符个数时，超出字符将被隐藏，单击“展开”按钮可查看全部内容。 **说明** 默认开启自动折叠长日志，字符个数默认为400个。
日志时间展示	鼠标悬浮在 按钮上，单击“日志时间展示”，在弹出的日志折叠设置页面中，设置是否展示毫秒、是否展示时区。 **说明** 默认开启展示毫秒。
虚拟滚动设置	鼠标悬浮在 按钮上，单击“虚拟滚动设置”，在弹出的虚拟滚动设置页面中，设置是否开启虚拟滚动、填写缓冲区大小。 **说明** ● 虚拟滚动可以避免或减少滚动时卡顿的情况，提升操作体验，防止页面卡死。 ● 滚动时数据会重新渲染，一定程度上影响数据流畅性。 ● 缓冲区决定同时加载的数据量大小，缓冲区越大，同时加载的数据越多，但滚动性能会越差。
不可见字段列表	该列表展示版面设置中配置的不可见性字段。 ● 当日志流未配置版面设置时，将不显示 按钮。 ● 当日志内容为“CONFIG_FILE”且未配置版面设置时，不可见字段默认有appName、clusterId、clusterName、containerName、hostIPv6、NameSpace、podName和serviceID。

5.4.2 LTS 搜索语法介绍

云日志服务LTS提供一套搜索语法用于设置搜索条件，用于指定日志查询时的过滤规则，从而筛选日志中满足条件的记录，筛选结果可以用于分析语句，进行更复杂的分析处理。

说明

- 使用搜索语法前，请您在索引配置处设置对应分词符，如无特殊需要，可直接使用默认的分词符，**'";=()[]{}@&<>/:\n\t\r。**
- 搜索语法不支持对分词符进行搜索。

  搜索语句不支持区分分词符，例如搜索语句**var/log**，其中/为分词符，搜索语句等同于**var log**，搜索的是同时包含**var**和**log**的所有日志。同理，搜索语句**"var:log"**、**var;log**等搜索的也是同时包含**var**和**log**的所有日志。
- 查询日志使用搜索语法的常见问题和相关报错的处理方法请参考**日志搜索相关问题**。

搜索方式

搜索语句用来指定日志搜索时的过滤规则，返回符合条件的日志。

根据索引配置方式可分为全文搜索和字段搜索，根据搜索精确程度可分为精确搜索和模糊搜索。其他类型的搜索方式包括范围搜索、短语搜索等。

表 5-16 搜索方式说明

搜索方式	说明	示例
全文搜索	配置全文索引后，日志服务根据您设置的分词符将整条日志拆分成多个关键词。 **说明** • content为日志原文对应的内置字段，搜索语句GET等同于content:GET，默认匹配日志原文的内容。 • 多个关键词默认通过AND连接，搜索语句GET POST等同于GET and POST。	• GET POST • GET and POST • content:GET and content:POST 上述三个搜索语句功能相同，均表示搜索同时包含关键词GET和POST的日志。

搜索方式	说明	示例
字段搜索	配置字段索引后，您可以指定字段名称和字段值（key:value）进行搜索。根据字段索引中设置的数据类型，您可以进行多种类型的基础搜索和组合搜索。 **说明** • value参数不可为空，通过搜索语句**key:""**匹配字段值为空的日志。 • 字段搜索和 not 运算符配合使用时，还会匹配到不包含该字段的日志。	• request_time>60 and request_method:po*表示搜索request_time字段值大于60且request_method字段值以po开头的日志。 • request_method:""表示搜索request_method字段值为空的日志。 • not request_method:GET表示搜索不包含request_method字段和request_method字段值不为GET的日志。
精确搜索	使用精确的词进行搜索。 日志服务搜索采用的是分词法，搜索时不会保证关键词出现的顺序。 **说明** 搜索语句为abc def，会匹配所有同时包含abc和def的日志，日志abc def或者def abc都会命中，如果需要确保关键词出现的顺序，请您采用**短语搜索**#**"abc def"**。	• GET POST表示搜索同时包含关键词GET和POST的日志。 • request_method:GET表示搜索request_method字段值包含GET的日志。 • #"/var/log"表示搜索包含短语/var/log的日志。
模糊搜索	在搜索语句中指定一个词，在词的中间或者末尾加上模糊搜索关键字，即星号（*）或问号（?），日志服务会在所有日志中搜索到符合条件的词，返回包含这些词并满足搜索条件的所有日志。 **说明** • 星号（*）代表匹配多个字符，问号（?）代表匹配1个字符。 • 星号（*）或问号（?）不能用在词的开头。 • long数据类型和float数据类型不支持使用星号（*）或问号（?）进行模糊搜索。	• GE*表示在所有日志中查找以GE开头的词，并返回包含这些词的日志。 • request_method:GE*表示在所有日志中查找request_method字段值以GE开头的词，并返回包含这些词的日志。

搜索方式	说明	示例
范围搜索	long数据类型和float数据类型支持范围搜索。 ● 方式1：通过 =（等于）>（大于）<（小于）运算符搜索日志。 ● 方式2：通过 in 运算符搜索日志，支持修改开闭区间。 **说明** string类型的字段不支持范围查询。	● request_time>=60表示在所有日志中查找request_time字段值大于等于60的日志。 ● request_time in (60 120]表示在所有日志中查找request_time字段值大于60且小于等于120的日志。
短语搜索	短语搜索用于完全匹配日志中的目标短语，可以确保关键词出现的顺序。 **说明** 短语搜索不支持模糊搜索。	**#"abc def"**表示在所有日志中查找包含目标短语 abc def 的日志。

- 分词符

  云日志服务LTS会根据分词符，将日志内容拆分成多个词。日志服务默认配置的分词符为 **, '";=()[]{}@&<>/:\n\t\r**。

  例如日志**2023-01-01 09:30:00**，默认分词符会将其分为四部分：**2023-01-01**、**09**、**30**、**00**。

  此时搜索语句**2023**无法匹配到该条日志，可以通过**2023-01***或**2023-01-01**搜索到该条日志。

  如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整日志内容或模糊搜索查找对应的日志。

- 关键词顺序

  只有短语搜索**#"abc def"**才能保证关键词出现的顺序，其他搜索方式多个关键词默认AND连接。

  例如**request_method:GET POST**查询的是同时包含GET和POST的日志，不会保证GET和POST的顺序。如有需要推荐采用**短语搜索**。

- 中文搜索

  中文搜索时不需要采用模糊查询，如有需要推荐采用短语搜索，可以匹配到更精确的结果。

  云日志服务LTS的英文是以单词的形式进行拆分的，单词的长度不一致，因此可以通过模糊搜索匹配拥有相同前缀英文单词的日志。

  中文采用的是一元分词，每个字都是独立的，拆分后每部分的长度都是1。

  例如搜索语句**星期一**，代表搜索同时包含**星**、**期**、**一**的日志；搜索语句**#"星期一"**，代表搜索包含目标短语**星期一**的日志。

- 语法关键词

  日志搜索语句的语法关键词包括：&& || AND OR and or NOT not in : > < = ( ) [ ] 中文冒号 中文双引号

  其中 **and AND or OR NOT not in** 作为语法关键词使用时，前后需要使用空格分隔；

  如果日志中本身包含语法关键词且需要搜索时，搜索语句需要用**双引号**包裹，否则可能会导致语法错误或搜索到错误的结果。

例如搜索语句**content:and**, 包含语法关键词 **and** ,需要修改为**content:"and"**。

短语搜索

短语搜索用于准确匹配目标短语，例如搜索语句**abc def**，不区分先后顺序，将匹配所有同时包含**abc**和**def**的日志。短语搜索和关键词搜索的区别请参考**表5-17**。

- 短语搜索：在关键词搜索语法的基础上实现，短语搜索能够区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。短语搜索适用于英文短语、中文短语的搜索，不支持模糊搜索。
- 关键词搜索：关键词搜索是基于分词实现，通过分词符先将搜索内容拆分为多个关键词，然后匹配日志。关键词搜索不会区分多个关键词在日志中出现的顺序，因此只要日志中按照搜索的与或非逻辑能命中关键词，该日志就会被搜索到。

表 5-17 搜索区别

搜索方式	短语搜索	关键词搜索
搜索区别	区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。	不区分关键词的顺序，按照搜索逻辑命中关键词即可。
举例说明	假设您的日志流中存在2条原始日志，如下： • 原始日志1：this service is lts • 原始日志2：lts is service	
	短语搜索：#"is lts"，会命中1条日志。	关键词搜索：is lts，会命中2条日志。
	短语搜索：#"lts is"，会命中1条日志。	关键词搜索：lts is，会命中2条日志。

使用限制：

- 短语搜索不支持搭配模糊搜索。

  短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持搭配模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。
- 短语搜索不支持对分词符进行搜索。

  例如搜索语句**#"var/log"**，其中 / 为分词符，搜索语句等同于**#"var log"**，会搜索包含目标短语**var log**的日志。同理，搜索语句**#"var:log"**、**#"var;log"**等搜索的也是包含目标短语**var log**的日志。
- 中文搜索推荐采用短语搜索。

  由于中文默认采用的是一元分词，每个汉字单独分词，搜索时会匹配同时包含搜索语句中每一个汉字的日志，本身便具有模糊搜索的特性，当需要更加精确的结果时，推荐采用短语搜索。

运算符

搜索语句支持的运算符请参考**表5-18**。

说明

- 除in运算符外，其他运算符不区分大小写。
- 运算符的优先级由高到低排序如下所示：
  1. 冒号（:）
  2. 双引号（""）
  3. 圆括号( )
  4. and、not
  5. or

表 5-18 运算符说明

运算符	说明
and	与运算符，如果多个关键词之间没有语法关键词，默认为and关系，例如**GET 200**等同于**GET and 200**。 **说明** and作为运算符使用时前后需要使用空格分隔。例如 **1 and 2** 代表搜索同时包含**1** 和**2**的日志；**1and2**代表搜索包含词语**1and2**的日志。
AND	与运算符，等同于and。
&&	与运算符。 **说明** &&作为运算符使用时不需要使用空格分隔。例如 **1 && 2** 等同于**1&&2**，代表搜索同时包含**1** 和**2**的日志。
or	or运算符，例如**request_method:GET or status:200**。 **说明** or 作为运算符使用时前后需要使用空格分隔。
OR	或运算符，等同于or。
||	或运算符。|| 作为运算符使用时不需要使用空格分隔。
not	非运算符。例如**request_method:GET not status:200**、**not status:200**。 **说明** • not 作为运算符使用时需要使用空格分隔。 • not 运算符和字段搜索配合使用时还会匹配到不包含对应字段的日志。
( )	用于提高括号内搜索条件的优先级。例如**(request_method:GET or request_method:POST) and status:200**。
:	用于字段搜索（key:value），例如**request_method:GET**。 **说明** 如果字段名称（key）或者字段值（value）内有空格或冒号（:）等保留字符，请使用双引号（""）包裹字段名称（key）或者字段值（value）。例如： • **"request method":GET** • **message:"This is a log"** • **time:"09:00:00"** • **ipv6:"2024:AC8:2ac::d09"**

运算符	说明
""	使用双引号（""）包裹一个语法关键词，可以将该语法关键词转换成普通字符，例如"and"表示搜索包含and的日志，此处的and不代表运算符。
\	转义符号，用于转义双引号（""），转义后的引号表示符号本身。例如日志内容为instance_id:nginx"01"，您可以使用instance_id:nginx\"01\"进行查询。
*	通配符搜索，匹配零个、单个、多个字符。例如**request_method:P*T**。 **说明** 不支持放在关键词开头，推荐放在关键词的中间部分或者结尾。
?	通配符搜索，匹配单个字符。例如**request_method:P?T**，可以匹配到PUT，无法匹配到POST。 **说明** 不支持放在关键词开头，推荐放在关键词的中间部分或者结尾。
>	搜索某字段值大于某数值的日志。例如**request_time>100**。
>=	搜索某字段值大于或等于某数值的日志。例如**request_time>=100**。
<	搜索某字段值小于某数值的日志。例如**request_time<100**。
<=	搜索某字段值小于或等于某数值的日志。例如**request_time<=100**。
=	搜索某字段值等于某数值的日志，仅适用于float、long类型的字段。对于该类型的字段，等号（=）和冒号（:）作用相同。例如**request_time=100**等同于**request_time:100**。
in	搜索某字段值处于某数值范围内的日志，中括号表示闭区间，小括号表示开区间，两个数字之间使用空格分隔。例如**request_time in [100 200]**或**request_time in (100 200]**。 **说明** in只能为小写字母，且作为运算符使用时前后需要使用空格分隔。
#""	用于搜索包含目标短语的日志，可以保证关键词出现的顺序。 **说明** 短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。

搜索语句示例

同一条搜索语句，针对不同的日志内容和索引配置时，会有不同的搜索结果。本文基于如下日志样例和索引介绍搜索语句示例。

表 5-19 普通搜索示例

搜索需求	搜索语句
搜索POST请求且状态码为200的日志。	request_method:POST and status=200
搜索GET请求或POST请求成功（状态码为200~299）的日志。	(request_method:POST or request_method:GET) and status in [200 299]
搜索GET请求或POST请求失败的日志。	(request_method:POST or request_method:GET) not status in [200 299]
搜索非GET请求的日志。	not request_method:GET
搜索GET请求成功且请求时间小于60秒的日志。	request_method:GET and status in [200 299] not request_time>=60
搜索请求时间为60秒的日志。	• request_time:60 • request_time=60
搜索请求时间大于等于60秒，并且小于200秒的日志。	• request_time>=60 and request_time<200 • request_time in [60 200)
搜索包含and的日志。	content:"and" **说明** 此处使用双引号将and包裹，and为普通字符串，不代表运算符。
搜索不存在user字段的日志。	not user:*
搜索user字段值为空的日志。	user:""

搜索需求	搜索语句
搜索星期字段值不为星期一的日志。	not week:星期一
搜索sec-ch-ua-mobile字段值为?0的日志。	sec-ch-ua-mobile:#"?0" **说明** 日志内容中包含*或?且需要搜索时，需要采用短语查询。

下面介绍进阶搜索示例。

表 5-20 模糊搜索

搜索需求	搜索语句
搜索包含以GE开头的词的日志。	GE*
搜索包含以GE开头，结尾只有一个字符的词的日志。	GE?
搜索request_method字段值包含以G开头的词的日志。	request_method:G*
搜索request_method字段值包含以P开头，以T结尾，中间还有单个字符的词的日志。	request_method:P?T
搜索request_method字段值包含以P开头，以T结尾，中间包含零个、单个或多个字符的词的日志。	request_method:P*T

基于分词符的搜索，例如User-Agent字段值为**Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36**。

- 设置分词符为空时，该字段值将被当成一个整体，则您使用**User-Agent:Chrome**搜索语句进行搜索时，无法搜索到日志。
- 设置分词符为**, '";=()[]{}?@&<>/:\n\t\r**后，该字段值会被拆分为**Mozilla**、**5.0**、**Windows**、**NT**、**10.0**、**Win64**、**x64**、**AppleWebKit**、**537.36**、**KHTML**、**like**、**Gecko**、**Chrome**、**113.0.0.0**、**Safari**、**537.36**。

  此时可以使用**User-Agent:Chrome**等搜索语句进行搜索。

  表 5-21 基于分词符的搜索

搜索需求	搜索语句
搜索User-Agent字段值中包含Chrome的日志。	User-Agent:Chrome

搜索需求	搜索语句
搜索User-Agent字段值中包含以Win开头的词的日志。	User-Agent:Win*
搜索User-Agent字段值中包含Chrome和Linux的日志。	User-Agent:"Chrome Linux"
搜索User-Agent字段值中包含Firefox或Chrome的日志。	User-Agent:Chrome OR User-Agent:Linux
搜索User-Agent字段值包含Chrome，但不包含Linux的日志。	User-Agent:Chrome NOT User-Agent:Linux

5.4.3 创建 LTS 快速分析

日志包含了系统性能及业务等信息，例如关键词ERROR的多少反应了系统的健康度，关键词BUY的多少反应了业务的成交量等，当您需要了解这些信息时，可以通过快速分析功能，指定查询日志关键词，LTS能够针对您配置的关键词进行统计，并生成指标数据，以便您实时了解系统性能及业务等信息。

说明

- 支持对前100000条日志进行分析。
  快速分析的目的是快速返回字段值的分布情况和变化趋势，并没有对全量数据进行分析，是一种采样结果。
- 支持通过查询时间和查询条件过滤日志进行分析。
  快速分析是对通过查询语句查询到的日志进行分析，当查询到的日志数目为零时，快速分析没有结果。
- 支持将快速分析生成查询语句。
  单击快速分析的某一行分析结果，可自动生成查询语句，查询日志并生成新的快速分析。
- 快速分析的字段长度最大为2000字节。
- 快速分析字段分布统计展示前100条数据。
- 在快速分析的字段中，当分析时间范围内未开启快速分析、字段不存在或字段值为null时，分析结果为null。
  - 当字段为string类型时，单击null添加到搜索框中将显示为字段 : "null"OR NOT 字段 : *。
  - 当字段为float或long类型时，单击null添加到搜索框中将显示为NOT 字段 : *。
  - 未开启快速分析期间，不会存储分析所用的列存数据，分析结果对应为null，此时查询日志没有实际含义，不一定能匹配到日志。

前提条件

快速分析的对象为结构化日志中提取的关键字段，创建快速分析前请先对原始日志进行**设置云端结构化解析日志**。

创建快速分析

可通过日志结构化打开“快速分析”按钮进行创建。也可通过如下步骤进行创建。

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 快速分析以日志流为单位，请在“日志管理”页面选择目标日志组和日志流。

步骤3 支持两种方式创建快速分析：

1. 单击 进入设置详情页面，在索引配置页签的字段索引下方，添加字段时开启快速分析。
2. 在云端结构化解析页签，开启自动配置索引和快速分析，默认是开启状态。开启后将使用结构化字段配置字段索引并打开快速分析。

步骤4 在“日志搜索”页签，单击“创建快速分析”，跳转到索引配置页面添加需要快速分析的字段。

步骤5 单击“确定”，快速分析创建完成。

说明

- abc 表示String类型字段。
- 1.2 表示float类型字段。
- 123 表示long类型字段。
- 快速分析的字段长度最大为2000字节。
- 快速分析字段展示前100条数据。
- 支持显示当前字段的采样数量。
- 单击 即可查看一键生成的图表展示，string类型的字段支持展示字段分布值统计和智能聚合时间折线图，long和float数值类型的字段只支持展示智能聚合时间折线图。单击图表即可进入详情页面。
- 单击字段分布值统计或智能聚合时间折线图，会自动跳转到可视化界面并生成对应的SQL查询语句进行查询，更加直观地展示字段值的分布和变化趋势。更多信息请参见**可视化**。

----结束

二次分析

快速分析功能支持对float类型和long类型的字段进行二次分析，具体如下：

快速统计最大项、最小项、平均值和总和，分别单击目标字段下的**Max**、**Min**、**Avg**、**Sum**，快速查找所有项中的最大项、最小项、平均值和总和。

5.4.4 保存 LTS 快速查询日志条件

当您需要重复使用某一关键字搜索日志时，可以将其设置为快速查询语句。

保存 LTS 快速查询日志条件

1. 在云日志服务控制台，单击“日志管理”。
2. 在日志组列表中，单击日志组名称前对应的 。
3. 在日志流列表中，单击日志流名称，进入日志详情页面。
4. 在“日志搜索”页签，单击 ，输入“快速查询名称”和“快速查询语句”。默认开启快速查询和快速查询（保存本地）。

图 5-3 创建快速查询

- 快速查询名称，用于区分多个快速查询语句。名称自定义，需要满足如下要求：
  - 只支持输入英文、数字、中文、中划线、下划线及小数点。
  - 不能以小数点、下划线开头或以小数点结尾。
  - 长度为1-64个字符。
- 快速查询语句，搜索日志时需要重复使用的关键字，例如“error*”。

5. 单击“确定”，完成快速查询条件的创建。在左侧导航栏的快速查询页签，即可查看到保存成功的语句。
   单击快速查询语句的名称，查看日志详情。

查看上下文

您可以通过本操作查看指定日志生成时间点前后的日志，用于在运维过程中快速定位问题。

1. 在日志详情页面的原始日志页签，单击可以查看上下文。
   在查看上下文结果中，可以查看该日志的前后若干条日志详细信息。

2. 在弹出的查看上下文页面中，查看日志上下文。

表 5-22 查看上下文日志功能介绍

功能	说明
查询行数	根据需要选择查询日志的行数。
高亮显示	输入需要高亮的字符串，回车确认，在日志内容中高亮显示。
过滤日志	输入需要过滤的字符串，回车确认，在日志内容中高亮显示。当高亮显示和过滤日志同时设置时，均可高亮显示。
显示字段	查看上下文，默认字段为content，单击“显示字段”选择查看其他字段的上下文。
更早	从当前位置往前查看设置**查询行数**的二分之一。例如：当查询行数设置为100时，单击“更早”则从当前位置朝前显示50行，此时行号为-50；再次单击“更早”，依次叠加分别为-100、-150、-200.....
当前位置	当前日志位置。当设置了更早或更新时，单击“当前位置”可回到查看上下文开始的位置，即行数为0时。
更新	从当前位置往后查看设置**查询行数**的二分之一。例如：当查询行数设置为100时，单击“更新”则从当前位置朝后显示50行，此时行号为50；再次单击“更新”，依次叠加分别为100、150、200.....
简洁模式	打开简洁模式，只显示行号和content内容。关闭简洁模式，显示日志详情。
下载	目前仅支持下载content字段内容到本地查看。

5.5 查看 LTS 实时日志

日志接入LTS后，日志每隔大约1分钟上报一次，在实时日志页面，您最多需要等待1分钟左右，即可查看实时上报的日志，实现对日志数据的快速检索与分析。

- 若实时日志大于1MB且总数小于2000条时时，LTS会清空前500KB的日志，保留最新的500KB日志。
- 若实时日志未超过1MB，但总数超过2000条时，LTS会清空前1000条日志，保留最新的1000条日志。

说明

正常情况下，每隔5秒加载一次。如果这5秒内没有产生日志，则不显示；5秒后会继续调用接口，刷新出产生的日志数据。即如果每5秒都有日志数据产生，则加载数据时延为5秒。

前提条件

- 已创建日志组和日志流。
- 已完成**ICAgent安装**。
- 已配置日志采集规则。

查看 LTS 实时日志

如果您正在使用实时查看功能，请停留在实时查看页面，请勿切换页面。如果离开实时查看页面，实时查看功能将会停止，重新开启后上一次查看的实时日志将不会显示。

步骤1 在云日志服务管理控制台，单击“日志管理”。

步骤2 在日志组列表中，单击日志组名称前对应的 。

步骤3 在日志流列表中，单击日志流名称，进入日志详情页面。

步骤4 在“实时日志”页签，查看实时日志。

说明

通过来源类型分别筛选主机和K8S的日志。

- 来源类型选择主机时， 设置主机IP和文件路径。
- 来源类型选择K8S时，设置实例名称、容器名称和文件路径。
- 字段过滤：从索引配置、结构化配置、最新日志获取。

日志每隔大约1分钟上报一次，在日志消息区域，您最多需要等待1分钟左右，即可查看实时上报的日志。

同时，还可以通过页面右上方的“清屏”、“暂停”对日志消息区域进行操作。

- 清屏：清除日志消息区域已经显示出来的日志。
- 暂停：暂停日志消息的实时显示，页面定格在当前已显示的日志。

  暂停后，“暂停”会变成“继续”，再次单击“继续”，日志消息继续实时显示。

----结束

5.6 分析 LTS 日志

可视化提供对结构化后的日志字段进行SQL查询与分析的功能。对原始日志结构化后，等待1~2分钟左右即可对结构化后的日志进行SQL查询与分析。

说明

目前此功能仅支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布二零一、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华南-深圳，支持部分白名单用户使用的局点有：亚太-曼谷、华北-北京一、华东-上海二、华北-乌兰察布二零二，其他局点暂不支持该功能。

前提条件

- 已成功采集到日志。
- 对日志内容已完成结构化配置，具体操作请参考**设置云端结构化解析日志**。

  说明

  日志结构化后字段名称与系统SQL内置保留字段名称相同，或者字段名称中带有中划线、下划线、小数点这三种特殊字符时，SQL查询需要加英文双引号。系统SQL内置保留字段名称包括："time"、"select"、"where"等。

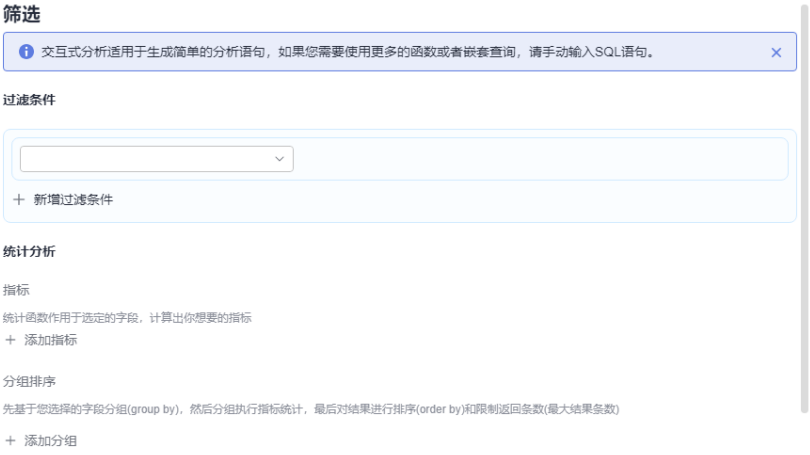
分析日志

步骤1 登录LTS控制台，在左侧导航栏中选择“日志管理 ”。

步骤2 在“日志管理”页面选择目标日志组和日志流，进入日志详情页面。

步骤3 选择“日志分析”页签。

步骤4 在可视化页面支持交互式分析，通过该模块配置简单的分析语句，查询可视化数据，配置可视化图表。设置过滤条件、通过添加指标、添加分组、添加排序进行数据通分析，方便用户操作。



步骤5 选择时间范围，输入SQL语句，单击 对已输入的SQL语句进行格式化显示。单击“查询”进行搜索，目前LTS支持的SQL语句详见**SQL分析语法介绍**。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

说明

- SQL查询约束有：
  1. 单次查询返回结果最多10W条。
  2. 当聚合结果超过10W时，聚合结果可能存有误差。
- SQL查询语句中，string类型的where条件的键值有限制：
  1. 精确查找value需添加英文单引号， 模糊查找value需添加英文单引号或者双引号，key与SQL内置保留字段名称相同时需添加英文双引号。
  2. 建议使用where条件时，使用where "key"='value'，或者where "key" like '%value%'。
- SQL查询语句中，float和long类型的where条件不受限制，但当与关键词冲突时可能会导致查询异常，建议使用where "key"='value'，或者where "key" like '%value%'进行查询。
- 日志搜索框支持自定义上下拖动调整高度。
- 输入搜索语法后，支持单击 设置格式化sql和反格式化sql，优化搜索语句，提高搜索效率。

步骤6 当设置时间范围内日志量超过10亿行时会触发迭代查询，可以通过迭代查询分多次完成全部日志的查询，界面会显示“查询状态：结果精确”。

步骤7 根据SQL查询返回的数据，依照业务需求选择不同图表类型，呈现查询结果。详细请参考**使用统计图表将日志可视化**。

步骤8 对查询结果可执行如下操作：

- 单击“新建”，在弹出的“创建可视化图表”中，根据业务需求填写“图表名称”、“同时添加到仪表盘”，单击“确定”，可视化图表保存成功。
- 单击“保存”，对在弹出的“保存可视化图表”中，根据业务需求填写“图表名称”、“同时添加到仪表盘”，单击“确定”，可视化图表保存成功；当选中某个可视化图表时，单击“保存”，可对该图表进行修改。
- 单击“另存为”，在弹出的“另存为可视化图表”中，根据业务需求填写“图表名称”、“同时添加到仪表盘”，单击“确定”，对已有可视化图表进行复制。

  说明

  须先保存一个图表后，才可另存为可视化图表。
- 单击“下载”，可下载当前SQL查询结果的可视化数据，该文件为.csv。
- 单击 按钮添加告警，在弹出的“新建告警规则”中，为选中的可视化图表配置**创建SQL告警规则**。

  说明

  须先保存一个图表后，才能新建告警规则。
- 单击“展开图表”，可对当前日志流下的可视化图表展开；单击“收起图表”，可收起当前日志流下展开的可视化图表。

----结束

5.7 SQL 分析语法介绍

5.7.1 SQL 查询语法概述

SQL是用于访问和处理数据库的标准计算机语言。LTS SQL提供了查询日志流中结构化数据的语句, 以下均将 LTS SQL 称为 SQL。

SQL语言由用于处理数据库和数据库对象的命令和函数组成。使用该语言时需遵循有关表达式和文本使用的规则。因此在SQL参考章节，除了SQL语法参考外，还会看到有关表达式、函数和操作符等信息。SQL基本查询语句如下：

说明

目前此功能仅支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布二零一、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华南-深圳，支持部分白名单用户使用的局点有：亚太-曼谷、华北-北京一、华东-上海二、华北-乌兰察布二零二，其他局点暂不支持该功能。

语法格式

```
SELECT [ ALL | DISTINCT ] { * | exprs }
FROM { <subquery>}
[ WHERE where_condition ]
[ GROUP BY [ col_name_list ]
[ HAVING expr ]
[ ORDER BY expr [ ASC | DESC ], expr [ ASC | DESC ], ... ]
[ LIMIT limit ]
[ OFFSET offset ]
```

数据类型

SQL查询中支持的数据类型如表5-23。如果当前字段数据类型需要改为其他数据类型，我们会进行数据类型的转换。例如STRING类型的字段转为LONG类型。字段数据类型转换之后的结果将会显示默认值，如STRING类型的数据转换为LONG类型的数据，结果会显示为LONG类型的默认值0。同理，当空值被转换为非空类型值时，也会使用默认值进行替换。例如，当把STRING类型空值转换为数字类型时，将会返回默认值0。

说明

SQL语法中，字符必须被单引号（''）包裹，无符号或双引号（""）包裹的为字段或表名称，如：'msg'表示字符串msg，msg或"msg"表示日志结构化msg字段。

表 5-23 SQL 查询支持的数据类型

原生数据类型	默认值	说明
STRING	""	原生STRING类型
FLOAT	0.0	原生FLOAT类型
LONG	0	原生LONG类型

查询语句

表 5-24 SQL 查询语句

语句	说明	示例
DISTINCT	返回去重后的结果。	SELECT DISTINCT visitCount
FROM	表示当前查询数据的源数据集， 可以是当前日志流的结构化数据, 也可以是当前日志流结构化数据的一个子集。 不加FROM的时候默认从当前日志流结构化数据查询，如果查询的数据源是一个子集， 则需要自己编写子查询语句。	SELECT visitCount
WHERE	指定查询的过滤条件，支持算术运算符、关系运算符和逻辑运算符。具体过滤条件可填在where_condition处。	SELECT visitCount WHERE visitCount > 0
GROUP BY	指定作为分组依据的结构化字段，支持根据单字段或多字段分组。具体的结构化字段列表可填入col_name_list处。	SELECT host, count(*) AS pv WHERE visitCount > 0 GROUP BY host
HAVING	只能与GROUP BY配合使用。指定用于过滤GROUP BY结果的结构化字段。	SELECT host, count(*) AS pv GROUP BY host HAVING pv > 10
ORDER BY	后面的字段必须是用于GROUP BY分组的字段，对GROUP BY的查询结果进行排序，用于排序的可以是任意一个结构化字段。	SELECT host, count(*) AS pv GROUP BY host ORDER BY pv
ASC/DESC	ASC为升序，DESC为降序，默认为ASC。	SELECT host, count(*) AS pv GROUP BY host ORDER BY pv DESC
LIMIT	对查询结果进行限制，用于限制返回的结构化日志条数。一次查询最多返回100000条结构化日志。 **说明** 如果不使用LIMIT语句，默认返回查询结果中最新的100条数据。	SELECT host LIMIT 100

示例

表 5-25 常用 SQL 查询语句示例

查询需求	查询语句
标准查询	SELECT "field" WHERE "field" = 'value'
统计行数	SELECT count(*)
列的别名	SELECT count(*) AS "pv"
去重查询	SELECT DISTINCT("field")
分页查询	SELECT "field" LIMIT 100
排序查询	SELECT "__time" ORDER BY "__time"
分组查询	SELECT "field" GROUP BY "field"
分组统计	SELECT "field",count(*) GROUP BY "field"
模糊查询	SELECT "field" LIKE 'value%'
查询总和	SELECT sum("field")
查询最大值	SELECT max("field")
查询最小值	SELECT min("field")
查询平均值	SELECT avg("field")
SQL嵌套子查询	SELECT sum(pv) FROM (SELECT "field",count(*) AS "pv" GROUP BY "field")
HAVING子句过滤	SELECT "field",count(*) AS "pv" GROUP BY "field" HAVING "pv" > 10
查询包含GET，POST请求	SELECT * WHERE "request_method" IN ('GET', 'POST')
查询不包含GET，POST请求	SELECT * WHERE "request_method" NOT IN ('GET', 'POST')
查询非GET请求的日志	SELECT * WHERE "request_method" != 'GET'
查询GET请求成功并且状态码为200且请求时间小于60秒的日志	SELECT * WHERE "request_method" = 'GET' AND "request_time" < 60
查询请求时间大于等于60秒，并且小于200秒的日志	SELECT * WHERE "request_ time" >=60 and "request_time" < 200
查询GET请求或POST请求的日志	SELECT * WHERE "request_method" = 'GET' OR "request_method" = 'POST'

下面的语句是根据ELB结构化日志构造出的查询语句， 它包含所有的基础查询语法，仅供参考。

```
SELECT url AS Url, host AS Host, failure_rate AS FailureRate,
CONCAT(CAST(access_count AS varchar), ' times') AS "All",
CONCAT(CAST(rsp_200_count AS varchar), ' times') AS "COUNT_200"
FROM ( SELECT
CONCAT(host, CASE WHEN STRPOS(router_request_uri, '?') = 0 THEN router_request_uri ELSE
SUBSTR(router_request_uri, 1, 1) END) AS url,
host,count(1) AS access_count,
SUM(CASE WHEN status = 200 THEN 1 ELSE 0 END) AS "rsp_200_count",
(CASE WHEN COUNT(1) < 30 THEN 0 ELSE round(SUM(CASE WHEN status >= 400 THEN 1 ELSE 0 END) *
100.0 / COUNT(1), 2) END) AS failure_rate
WHERE host NOT IN ('monitor-new.olayc.cn')
GROUP BY host,router_request_uri
HAVING router_request_uri NOT IN ('/robots.txt', '/null', '/undefined')
)
ORDER BY FailureRate DESC
LIMIT 100
```

5.7.2 SQL 聚合函数

聚合函数是对结构化后的日志的指定列进行的统计运算。聚合函数返回的是单个值，经常与SELECT语句和GROUP BY语句一起使用。LTS支持如下表格所示聚合函数，具体请参考**表5-26**。

在聚合函数的使用中请注意以下几点：

- 聚合函数可用在任何查询的SELECT子句中。您可以使用如AGG(expr) FILTER(WHERE whereExpr)的语法在聚合之前进行过滤，即聚合函数只会聚合满足过滤条件的列。
- 在同一个SQL查询语句中，根据过滤条件的不同，对应的聚合函数所呈现的结果会不同。
- 只有COUNT函数可以跟DISTINCT搭配使用。
- 聚合操作没有固定的执行顺序。如果在执行具有多个聚合函数的SQL语句查询时，执行聚合函数的顺序对运算结果有影响，即结果会因执行顺序的不同而不同，那么每次执行这个查询，得出的结果可能会不一致。
- 如果需要聚合的数据为FLOAT类型时，数次执行同一个查询可能也会因此得出不同的聚合结果。如果您希望执行同一个查询时都能得出同样的结果，建议使用ROUND函数来消除多次查询之间的不一致。

语法格式

```
SELECT COUNT(fieldname1)
```

聚合函数语句

表 5-26 聚合函数语句

语句	说明	示例
COUNT(*)	统计行数。	SELECT COUNT(*)
COUNT(DISTINCT expr)	统计字段中去重后的行数，字段值可以是字符串或者数字，返回值为估算值（**默认存在2.3%的标准误差**）。	SELECT COUNT(DISTINCT host)

语句	说明	示例
SUM(expr)	返回数字总和。	SELECT SUM(visitCount)
MIN(expr)	返回数字中的最小值。	SELECT MIN(visitCount)
MAX(expr)	返回数字中的最大值。	SELECT MAX(visitCount)
AVG(expr)	返回平均值。	SELECT AVG(visitCount)
EARLIEST(expr)	表达式必须是数值类型的， 返回expr的最早的值， 即查询的时候最先遇到的值。	SELECT EARLIEST(visitCount)
LATEST(expr)	表达式必须是数值类型的， 返回expr的最新的值， 即查询的时候最后遇到的值。	SELECT LATEST(visitCount)
APPROX_QUANTILE_DS(expr, probability)	计算数值expr的近似分位数，probability应介于0和1之间。	APPROX_QUANTILE_DS(expr, probability)

5.7.3 SQL 同比和环比函数

本文介绍同比和环比函数的基础语法和示例。

compare 函数

compare函数用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。

语法格式

- 对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。
  ```
  compare(x,n)
  ```
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。
  ```
  compare(x, n1, n2, n3...)
  ```

参数说明

表 5-27 同比函数参数说明

参数	说明
x	目标列的列名，参数值为double类型或long类型。
n	时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。

返回类型

JSON数组。格式为[当前计算结果,n秒前的计算结果,当前计算结果与n秒前计算结果的比值]。

示例说明

计算当前1小时和昨天同时段的访问量比值。

1. 选择查询和分析的时间范围为**1小时（整点时间）**，并执行如下查询和分析语句。其中**86400**表示当前时间减去86400秒(1天)。
```
SELECT
compare(PV, 86400)
FROM  (SELECT count(*) AS PV )
```
2. 查询和分析结果

图 5-4 查询和分析结果

EXPR$0
[5994.0,6000.0,0.999]

说明

- **5994.0**表示当前1小时（例如2021-01-02 00:00:00~2021-01-02 01:00:00）的网站访问量。
- **6000.0**表示昨天同时段（例如2021-01-01 00:00:00~2021-01-01 01:00:00）的网站访问量。
- **0.999**表示当前1小时与昨天同时段的网站访问量比值。

3. 分列显示查询和分析结果
```
SELECT
diff[1] as "today",
diff[2] as "yesterday",
diff[3] as "ratio"
FROM(SELECT compare(pv, 86400) AS diff FROM (SELECT count(*) AS pv ))
```

图 5-5 查询和分析结果

today	yesterday	ratio
5993	6029	0.99402887

ts_compare 函数

ts_compare函数用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。

说明

ts_compare函数必须按照时间列进行分组（GROUP BY）。

语法格式

- 对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。
```
ts_compare(x, n)
```
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。
```
ts_compare(x, n1, n2, n3...)
```

参数说明

表 5-28 环比函数参数说明

参数	说明
x	参数值为double类型或long类型。
n	时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。

返回类型

JSON数组。格式为[当前计算结果, n秒前的计算结果, 当前计算结果与n秒前计算结果的比值, n秒前的UNIX时间戳]。

示例说明

环比今天3小时与昨天3小时的网站访问量。

选择查询和分析的时间范围为今天某3小时，并执行如下查询和分析语句。其中86400表示当前时间减去86400秒（1天），date_trunc('hour',__time)表示使用date_trunc函数将时间对齐到小时。

- 查询和分析语句

```
SELECT
  t_time,
  ts_compare(PV, 86400) AS data
FROM(
    SELECT
      date_trunc('hour', __time) AS t_time,
      count(*) AS PV
    GROUP BY
      t_time
    ORDER BY
      t_time
  )
GROUP BY
  t_time
```

- 查询和分析结果

t_time	data
2021-10-26T06:00:00.000Z	[159.0,224.0,0.7098214285714286,1.6351416E9]
2021-10-26T07:00:00.000Z	[100.0,148.0,0.6756756756756757,1.6351452E9]
2021-10-26T08:00:00.000Z	[100.0,100.0,1.0, 1.6016544E9, 1.6351488E9]

5.7.4 SQLJSON 函数

功能描述

JSON函数用于解析JSON对象或JSON数组，并从中提取值。

语法格式

```
SELECT json_extract(Results, '$.[0].EndTime')
```

JSON 函数语句

表 5-29 JSON 函数语句

语句	说明	示例	返回值类型
json_extract	用于从JSON对象或JSON数组中提取一组JSON值（数组或对象）	json_extract(x, json_path)	JSON格式的string类型
json_extract_scalar	用于从JSON对象或JSON数组中提取一组标量值（字符串、整数或布尔值）。如果指定JSON路径下不是标量，则返回null。	json_extract_scalar(x,json_path)	varchar类型

示例及说明

- **json_extract函数**

  获取Results字段中EndTime字段的值。

  a. 字段样例
  ```
  Results:[{"EndTime":1626314520},{"FireResult":2}]
  ```
  b. 查询和分析语句
  ```
  SELECT json_extract(Results, '$.[0].EndTime')
  ```
  c. 查询和分析结果

  表 5-30 查询和分析结果

EXPR$0
1626314520

- **json_extract_scalar函数**

  从Results字段中获取RawResultCount字段的值，并将这些值转换为bigint类型进行求和。

  a. 字段样例
  ```
  Results:[{"EndTime":1626314520},{"RawResultCount":1}]
  ```
  b. 查询和分析语句
  ```
  SELECT sum(cast(json_extract_scalar(Results,'$.[1].RawResultCount') AS bigint) )
  ```
  c. 查询和分析结果

  表 5-31 查询和分析结果

EXPR$0
1546

5.7.5 SQLIP 函数

使用限制

LTS提供的IP与地域之间的关系，来自于第三方IP库，且数据是周期性更新（约半年），不承诺IP与地域关系完全正确；后续LTS会优化，缩短IP库的更新周期，为用户提供更好的体验。

单IP函数聚合查询的数据量上限为500万，查询的数据超出上限可能导致查询超时。

功能描述

IP函数是分析目标IP地址所属的国家、省份、城市及对应的网络运营商。

语法格式

```
SELECT count(*) AS PV, ip_to_province(client_ip) AS province GROUP BY province
```

IP 函数语句

表 5-32 IP 函数语句

语句	说明	示例
ip_to_province	分析目标IP地址所属省份。	ip_to_province(x)
ip_to_country	分析目标IP地址所属国家或地区。	ip_to_country(x)
ip_to_city	分析目标IP地址所属城市。	ip_to_city(x)
ip_to_provider	分析目标IP地址所对应的网络运营商。	ip_to_provider(x)
ip_to_geo	根据传入的ip返回ip所在的经纬度。	ip_to_geo(x)

示例及说明

- **ip_to_province函数**

  统计请求总数Top3的省份

  a. 查询和分析语句

  ```
  SELECT count(*) AS PV, ip_to_province(client_ip) AS province GROUP BY province ORDER BY PV desc LIMIT 3
  ```

  b. 查询和分析结果

  表 5-33 查询和分析结果

PV	province
101	广东

PV	province
83	上海
78	山东

- **ip_to_country函数**

  统计请求总数的Top3的国家或地区

  a. 查询和分析语句

  ```
  SELECT count(*) AS PV, ip_to_country(client_ip) AS county GROUP BY country ORDER BY PV desc LIMIT 3
  ```

  b. 查询和分析结果

  表 5-34 查询和分析结果

PV	country
100	中国
76	美国
55	加拿大

- **ip_to_city函数**

  统计请求总数的Top3的城市

  a. 查询和分析语句

  ```
  SELECT count(*) AS PV, ip_to_city(client_ip) AS city GROUP BY city ORDER BY PV desc LIMIT 3
  ```

  b. 查询和分析结果

  表 5-35 查询和分析结果

PV	city
109	广州
89	上海
23	西安

- **ip_to_provider函数**

  统计请求总数的Top3的运营商

  a. 查询和分析语句

  ```
  SELECT count(*) AS PV, ip_to_provider(client_ip) AS provider GROUP BY provider ORDER BY PV desc LIMIT 3
  ```

  b. 查询和分析结果

表 5-36 查询和分析结果

PV	provider
115	电信
65	att.com
44	联通

- **ip_to_geo函数**
  根据传入的ip返回经纬度。
  a. 查询和分析语句
  ```
  SELECT count(*) AS PV, ip_to_geo (client_ip) AS geo GROUP BY province ORDER BY PV desc LIMIT 3
  ```
  b. 查询和分析结果

表 5-37

PV	geo
101	*, *
83	47.369013, -68.326674
78	32.715891, -117.161588

5.7.6 SQL 数学函数

功能描述

数学函数为标量函数中的一种，只支持数值类型的字段，能够实现对数值进行取整、取绝对值、求余等功能，具体请参考表5-38。

在数学运算中，如果表达式里涉及的操作数皆为整数，那么SQL将会采用整数运算，否则便会切换到浮点运算。您可以将其中一个操作数转换为FLOAT类型来强制进行切换，运行时SQL会将大多数表达式中的32位浮点数扩展到64位。

语法格式

```
SELECT ABS(fieldname1) AS fieldname1_abs
```

数学函数语句

表 5-38 数学函数语句

语句	说明	示例
ABS(expr)	取绝对值。	SELECT ABS(fieldname1)
CEIL(expr)	向上取整，即向上取最接近的整数值	SELECT CEIL(fieldname1)

语句	说明	示例
FLOOR(expr)	向下取整，即向下取最接近的整数值。	SELECT FLOOR(fieldname1)
TRUNCATE(expr, digits)	将expr截断为特定的digits位数。如果数字为负数，则会截断小数点左侧的许多位置。如果未指定，数字默认为零。	SELECT TRUNCATE(fieldname1, 2)
ROUND(expr, digits)	ROUND(expr, digits)对expr值进行四舍五入，保留小数位数由digits指定。expr可以是整数或浮点数，但digits必须是整数。返回值的类型由expr的类型决定。如果没有指定digits，则使用默认值0。如果digits是负数，则返回expr四舍五入后的整数。当expr是非数字值时，会被转换为数字0。如果expr是无限位数的数字，则被转换为最接近的DOUBLE类型的有限位数数字。	SELECT ROUND(fieldname1, 2)
x + y	加法。	SELECT fieldname1 + fieldname2
x - y	减法。	SELECT fieldname1 - fieldname2
x * y	乘法。	SELECT fieldname1 * fieldname2
x / y	除法。	SELECT fieldname1 / fieldname2
MOD(x, y)	求余，即取x除以y后的余数。	SELECT MOD(fieldname1, fieldname2)
LN(expr)	对数(以e为底)。	SELECT ln(expr)
LOG10(expr)	对数(以10为底)。	SELECT LOG10(expr)
POWER(expr,power)	expr的power次幂。	SELECT POWER(expr ,2)
SQRT(expr)	expr的平方根	SELECT SQRT(expr)
SIN(expr)	正弦	SELECT SIN(expr)
COS(expr)	余弦	SELECT COS(expr)
TAN(expr)	正切	SELECT TAN(expr)

语句	说明	示例
COT(expr)	余切	SELECT COT(expr)
ASIN(expr)	反正弦	SELECT ASIN(expr)
ACOS(expr)	反余弦	SELECT ACOS(expr)
ATAN(expr)	反正切	SELECT ATAN(expr)

示例及说明

ACOS(expr)函数

求参数值的反余弦，y=arccosx，x的取值范围[-1,1]。

1. 字段样例
   x:0.5
2. 查询和分析语句
   ```
   select  ACOS(x)
   ```
3. 查询和分析结果

   表 5-39 查询和分析结果

x	EXPR$1
0.5	1.0471975511965979

ATAN(expr)函数

ATAN求参数值的反正切，y= arctanx，x的取值范围R。

1. 字段样例
   x:0.5
2. 查询和分析语句
   ```
   select  ATAN(X)
   ```
3. 查询和分析结果

   表 5-40 查询和分析结果

x	EXPR$1
0.5	1.0471975511965979

ATAN2(expr)函数

ATAN2从直角坐标（x，y）到极坐标（r，θ）的转换角度θ。

1. 字段样例
   x:3；y:4

2. 查询和分析语句
```
SELECT x , y, ATAN2(x,y)
```
3. 查询和分析结果

表 5-41 查询和分析结果

x	y	EXPR$0
3	4	0.6435011087932844

5.7.7 SQL 时间函数

功能描述

时间函数可以与__time一起使用，任何存储为毫秒时间戳的列都可以使用 MILLIS_TO_TIMESTAMP 函数，或者任何存储为字符串时间戳的列都可以使用 TIME_PARSE 函数。默认情况下，时间操作使用UTC时区，您可以通过参数"timezone"设置为另一个时区的名称（如"Asia/Shanghai"）或设置为偏移量（如"+08:00"）来更改时区。

语法格式

语句	说明	示例
CURRENT_DATE	在连接时区的当期日期。	SELECT CURRENT_DATE
CURRENT_TIMESTAMP	在连接时区的当前时间戳。	SELECT CURRENT_TIMESTAMP
DATE_TRUNC(<unit>,<expr>)	截断时间戳，将其作为新时间戳返回。	SELECT DATE_TRUNC('minute',__time)
TIME_FORMAT(<expr>,<patten>,<timezone>))	使用给定的 pattern 或 ISO8601（例如 2000-01-02T03:04:05Z）将字符串解析为时间戳。timezone（如果提供）应为时区名称，如"America/Los_Angeles"或偏移量，如"+08:00"，并将用作不包括时区偏移量的字符串的时区。模式和时区必须是字面量。无法解析为时间戳的字符串将返回空值。	SELECT TIME_FORMAT(__time,'yyyy-MM-dd HH:mm:ss','+08:00')

语句	说明	示例
TIME_PARSE(<expr>,<pattern>,<timezone>)	使用给定的 pattern 或 ISO8601（例如 2000-01-02T03:04:05Z）将字符串解析为时间戳。时区（如果提供）应为时区名称，如 "America/Los_Angeles"或偏移量，如"+08:00"，并将用作不包括时区偏移量的字符串的时区。模式和时区必须是字面量。无法解析为时间戳的字符串将返回空值。	SELECT TIME_PARSE("timestamp",'yyyy-MM-dd HH:mm:ss','+08:00')
MILLIS_TO_TIMESTAMP(expr)	将时间戳转换为时间格式。	SELECT MILLIS_TO_TIMESTAMP(expr)
TIMESTAMP_TO_MILLIS(expr)	将时间转换为时间戳格式	SELECT TIMESTAMP_TO_MILLIS (expr)
EXTRACT(<extract_unit>FROM expr)	从expr中提取时间部分，并将其作为数字返回。	SELECT EXTRACT(MINUTE FROM __time)
TIMESTAMPDIFF(<unit>,<expr1>,<expr2>)	返回expr1和expr2之间的unit	SELECT TIMESTAMPDIFF (minute, expr1, expr2)
TIME_SERIES	补全您查询时间窗口内缺失的数据。	TIME_SERIES(__time, period, time_format, [padding_value],<timezone>)

TIME_SERIES 函数

time_series函数用于补全您查询时间窗口内缺失的数据。

说明

- 必须搭配ORDER BY语法使用，并且time_series函数是ORDER BY的第一个参数
- 查询语句中不能使用OFFSET语句
- time_series函数不支持作为子查询使用

语法格式

```
time_series(__time, period, time_format, [padding_value], <timezone>)
```

参数说明

表 5-42

参数	说明
__time	时间序列。
period	ISO 8601标准的时间窗口大小。例如P1M（1月）、P1D（1天）、PT1H（1小时）、PT1M（1分钟）、PT1S（1秒钟）。
time_format	返回结果的时间格式。（参考Joda DateTimeFormat模式）。
padding_value	补全的内容。包括： • 0 或 zero：将缺失的值设置为0（默认值）。 • null：将缺失的值设置为null。 • last：将缺失的值设置了上一个时间点对应的值。 • next：将缺失的值设置了下一个时间点对应的值。 • avg：将缺失的值设置为前后两个时间点的平均值。
timezone	时区。例如：北京时区：+08:00

返回类型

bigint类型。

示例说明

按照一天的时间粒度进行数据补全，将缺失的值设置为0，并添加时区。

- 查询和分析语句

```
select time_series(__time, 'P1D', 'yyyy-MM-dd HH:mm:ss', '0', '+08:00') as t_time, count(*) as num
group by t_time order by t_time
```

- 查询和分析结果

t_time	num
2021-10-01 08:00:00	5
2021-10-02 08:00:00	0
2021-10-03 08:00:00	0
2021-10-04 08:00:00	21
2021-10-05 08:00:00	17
2021-10-06 08:00:00	0
2021-10-07 08:00:00	34

CURRENT_DATE/ CURRENT_TIMESTAMP 函数

CURRENT_DATE返回查询当天的凌晨零点的ISO8601时间，返回的为UTC时间，该函数可直接参与时间戳之间的运算。

CURRENT_TIMESTAMP返回查询当前的ISO8601时间，返回的为UTC时间，该函数可直接参与时间戳之间的运算。

1. 字段样例

   __time: 2023-02-14T02:35:56.706Z

2. 查询和分析语句

```
select __time,CURRENT_DATE, CURRENT_TIMESTAMP,CURRENT_TIMESTAMP
```

3. 查询和分析结果

表 5-43 查询和分析结果

__time	CURRENT_DATE	CURRENT_TIMESTAMP
2023-02-14T02:35:56.706Z	2023-02-14T00:00:00.000Z	2023-02-14T14:35:57.000Z

DATE_TRUNC(<unit>, <timestamp_expr>)函数

舍去时间戳<timestamp_expr>中精度大于所选单位<unit>的值，将其置为零，并作为新时间戳返回。单位可以是'milliseconds'（毫秒）, 'second'（秒）, 'minute'（分）, 'hour'（时）, 'day'（日）, 'week'（周）, 'month'（月）, 'quarter'（季）, 'year'（年）, 'decade'（十年）, 'century'（千年）,'millennium'（世纪）, unit不区分大小写。

1. 字段样例

   __time: 2023-02-14T02:35:56.706Z

2. 查询和分析语句

```
SELECT __time,DATE_TRUNC('minute', __time),DATE_TRUNC('day', __time),DATE_TRUNC('year', __time)
```

3. 查询和分析结果

表 5-44 查询和分析结果

__time	EXPR$1	EXPR$2	EXPR$3
2023-02-14T02:35:56.706Z	2023-02-15T08:50:00.000Z	2023-02-15T00:00:00.000Z	2023-01-01T00:00:00.000Z

TIME_PARSE(<string_expr>, [<pattern>, [<timezone>]])/ TIME_FORMAT(<timestamp_expr>, [<pattern>, [<timezone>]])函数

TIME_PARSE将给定的字符串<timestamp_expr>，依据用户自定义的<pattern>参数以**Joda DateTimeFormat**模式解析为时间戳。若<pattern>未填写则以默认的ISO8601将字符串解析。<timezone>为时区，可省略。

TIME_FORMAT将给定的时间戳< timestamp_expr>，依据用户自定义的<pattern>参数以 **Joda DateTimeFormat**模式解析为字符串。若<pattern>未填写则以默认的ISO8601将时间戳解析。<timezone>为时区，可省略。

1. 字段样例

   __time: 2023-02-16T07:38:25.306Z

   start_time:2023-02-14 02:35:56

2. 查询和分析语句

```
SELECT __time,TIME_PARSE(start_time,'yyyy-MM-dd HH:mm:ss'),TIME_FORMAT(__time,'yyyy-MM-dd HH:mm:ss')
```

3. 查询和分析结果

表 5-45 查询和分析结果

__time	EXPR$1	EXPR$2
2023-02-16T07:38:25.306Z	2023-02-14T02:35:56.000Z	2023-02-16 07:38:25

MILLIS_TO_TIMESTAMP(millis_expr)/ TIMESTAMP_TO_MILLIS(timestamp_expr)函数

MILLIS_TO_TIMESTAMP函数将毫秒值转化为ISO8601格式的时间戳，转化后的参数可进行时间戳之间的运算。TIMESTAMP_TO_MILLIS将时间戳转化为毫秒值。

1. 字段样例

   __time: 2023-02-16T07:54:15.106Z，start_time: 1676534055106

2. 查询和分析语句

```
SELECT __time,MILLIS_TO_TIMESTAMP(start_time),TIMESTAMP_TO_MILLIS(__time)
```

3. 查询和分析结果

表 5-46 查询和分析结果

__time	EXPR$1	EXPR$2
2023-02-16T07:54:15.106Z	2023-02-16T07:54:05.000Z	1676534055106

TIME_EXTRACT(<timestamp_expr>,[<unit>,[<timezone>]])/ EXTRACT(<unit> FROM timestamp_expr)函数

TIME_EXTRACT函数，从timestamp_expr中提取时间部分，并将其作为数字返回。单位可以是EPOCH（返回纪元以来的秒值unix）、SECOND（当前分钟的秒值）、MINUTE（当前小时中的分钟值）、HOUR（当天的小时值）、DAY（当月的天数）、DOW（当前周的天数）、DOY（当前年的天数）、WEEK（当前年的周数）、MONTH（当前的月数）、QUARTER（当年的季度）或YEAR（返回当前年份）。<timezone>为时区，可省略。EXTRACT函数为TIME_EXTRACT的简写形式。

1. 字段样例

   __time: 2023-02-16T07:54:15.106Z，start_time: 1676534055106

2. 查询和分析语句
```
SELECT __time,MILLIS_TO_TIMESTAMP(start_time),TIMESTAMP_TO_MILLIS(__time)
```
3. 查询和分析结果

表 5-47 查询和分析结果

__time	EXPR$1	EXPR$2
2023-02-16T07:54:15.106Z	2023-02-16T07:54:05.000Z	1676534055106

参考信息

- **unit说明**

unit	说明
second	秒
minute	分
hour	时
day	日
week	周
month	月
quarter	季
year	年

- **extract_unit说明**

extract_unit	说明
SECOND	秒
MINUTE	分
HOUR	时
DAY	每月的第几天
DOW	每周的第几天
DOY	每年的第几天
WEEK	每年的第几周
MONTH	月
QUARTER	季
YEAR	年

5.7.8 SQL 最值函数

功能描述

SQL提供最值函数，对字段进行最值求解，具体请参见表5-48

最值函数对零个或多个字段进行操作，并返回单个值。

在最值函数的使用中请注意以下几点：

- 如果没有设置字段，则返回空值。字段必须能够转换为常见的数据类型。
- 如果所有字段都为空值，则返回空值。如果只有部分字段为空值，这些字段会被忽略。
- 如果字段中既有数字也有字符串，则函数将它们作为字符串进行比较。
- 如果所有字段都是整数，则函数将它们作为LONG值进行比较。
- 如果所有字段都是数字且至少有一个是FLOAT值，则函数将它们作为FLOAT值进行比较。

语法格式

```
SELECT GREATEST(fieldname1,fieldname2) AS the_greatest_field
```

最值函数语句

表 5-48 最值函数语句

语句	说明	示例
GREATEST([expr1, ...])	返回零个或多个字段间的最大值。	SELECT GREATEST(fieldname1,fieldname2)
LEAST([expr1, ...])	返回零个或多个字段间的最小值。	SELECT LEAST(fieldname1,fieldname2)

5.7.9 SQL 字符串函数

功能描述

SQL提供字符串函数，用于对字符类型的数据执行拼接、大小写转换等操作，具体请参见表5-49。

说明

SQL语法中，字符必须被单引号（''）包裹，无符号或双引号（""）包裹的为字段或表名称，如：'msg'表示字符串msg，msg或"msg"表示日志结构化msg字段。

语法格式

```
SELECT (fieldname1 || fieldname2) AS fieldname1_fieldname2
```

字符串函数语句

表 5-49 字符串函数语句

语句	说明	示例
CONCAT(expr1, expr2...)	拼接列举的所有字符串。	SELECT str1, str2, str3, CONCAT(str1, str2, str3) WHERE str1 IS NOT NUll
TEXTCAT(expr, expr)	拼接两个字符串。	SELECT str1, str2, TEXTCAT(str1, str2) WHERE str1 IS NOT NUll
STRING_FORMAT(pattern[, args...])	根据JAVA的String格式对字符串进行格式化。	SELECT str1, STRING_FORMAT(str1, '%s') WHERE str1 IS NOT NUll
LENGTH(expr)	返回字符串的长度，即字符串中UTF-16字符个数。	SELECT LENGTH(str1) WHERE str1 IS NOT NUll
LOWER(expr)	将字符串转换为小写形式。	SELECT LOWER(str1) WHERE str1 IS NOT NUll
POSITION(string1 IN string2 [FROM fromIndex])	返回string1在string2中首次出现位置的索引。搜索从指定索引开始，如果没有指定索引，则从索引1开始。如果string1不存在于string2中，则返回0。	SELECT POSITION(str1 IN str2 FROM 5)
REGEXP_EXTRACT(expr, pattern, [index])	返回字符串中匹配指定正则表达式的子字符串。索引从1开始。如果没匹配，则返回空值。如果没有指定索引，或者索引为0，则返回第一个匹配的子字符串。如想精确匹配，请在正则表达式前后分别加上符号^和$。	SELECT REGEXP_EXTRACT(str1, '[A-Za-z]+://[A-Za-z0-9.-]+(/[^]*)', 5)
REGEXP_LIKE(expr, pattern)	判断字符串是否匹配指定的正则表达式。如想精确匹配，请在正则表达式前后分别加上符号^和$。该函数与LIKE语句用法类似，区别在于LIKE语句搜索的是匹配指定模式的内容。	SELECT REGEXP_LIKE(str1, '\.(jpg|jpeg|png|gif)$')
REPLACE(expr, pattern, replacement)	使用replacement替换expr中与pattern相同的子字符串。	SELECT REPLACE(expr,pattern, replacement)

语句	说明	示例
STRPOS(string1, string2)	返回string2在string1中首次出现位置的索引。查找从索引1开始。如果查找没有结果，则返回0。	SELECT STRPOS(str1, str2) WHERE str1 IS NOT NUll AND str2 IS NOT NUll
SUBSTRING(expr, index, [length])	截取字符串。从指定索引处开始截取，结束位置由指定长度决定。长度按照UTF-16字符个数计算。	SELECT SUBSTRING(str1, 3, 10) WHERE str1 IS NOT NUll
RIGHT(expr, [length])	从字符串最右处开始往左截取指定长度。	SELECT RIGHT(str1, 5) WHERE str1 IS NOT NUll
LEFT(expr, [length])	从字符串最左处开始往右截取指定长度。	SELECT LEFT(str1, 5) WHERE str1 IS NOT NUll
SUBSTR(expr, index, [length])	与SUBSTRING相同。	SELECT SUBSTR(str1, 3, 10) WHERE str1 IS NOT NUll
UPPER(expr)	将字符串转换为大写形式。	SELECT UPPER(str1) WHERE str1 IS NOT NUll
REVERSE(expr)	反转字符串。	SELECT REVERSE(str1) WHERE str1 IS NOT NUll
LPAD(expr, length, chars)	在字符串左侧填充指定字符，直至字符串达到指定长度。如果指定长度小于字符串本身的长度，则按照指定长度对字符串执行截断操作。如果字符串或指定字符为空值，则返回空值。如果指定字符为空白，不会执行填充操作，但如有必要可能会删减字符。	SELECT LPAD(str1, 50, 'testStr') WHERE str1 IS NOT NUll
RPAD(expr, length, chars)	在字符串右侧填充指定字符，直至字符串达到指定长度。如果指定长度小于字符串本身的长度，则按照指定长度对字符串执行截断操作。如果字符串或指定字符为空值，则返回空值。如果指定字符为空白，不会执行填充操作，但如有必要可能会删减字符。	SELECT RPAD(str1, 50, 'testStr') WHERE str1 IS NOT NUll
CONTAINS_STRING(<expr>, str)	判断expr是否包含str字符串	SELECT CONTAINS_STRING(log_level,'warn')

语句	说明	示例
ICONTAINS_STRING(<expr>, str)	判断expr是否包含str字符串，不区分字符串大小写	SELECT ICONTAINS_STRING(log_level,'WARN')

示例及说明

REPEAT函数

REPEAT(expr, [N])函数将expr重复N次。

1. 字段样例

   field4:is

2. 查询和分析语句

```
select field4,REPEAT(field4,3)
```

3. 查询和分析结果

表 5-50 查询和分析结果

field4	EXPR$1
Is	isisis

5.7.10 SQL SPLIT 函数

功能描述

SPLIT函数用于通过指定的分隔符拆分字符串，并返回拆分后的子串集合。

语法格式

```
SELECT split_to_map(x, delimiter01, delimiter02)
```

SPLIT 函数语句

语句	说明	示例	参数
split	split函数用于通过指定的分隔符拆分字符串，并返回拆分后的子串集合。	split(x, delimeter,[limit])	• x：参数值为varchar类型。 • delimeter：分隔符。 • limit：限制字符串拆分的个数，大于0的整数。

语句	说明	示例	参数
split_part	split_part函数通过指定的分隔符拆分字符串，并返回指定索引的内容。	split_part(x, delimeter, part)	• x：参数值为varchar类型。 • delimeter：分隔符。 • part：指定要返回字段的索引值。
split_to_map	split_to_map函数用于使用指定的第一个分隔符拆分字符串，然后再使用指定的第二个分隔符进行第二次拆分。	split_to_map(x, delimiter01, delimiter02)	• x：参数值为varchar类型。 • delimeter01：分隔符1。 • delimeter02：分隔符2。

示例及说明

- **split函数**

  将目标字符串按指定字符串分割，limit用于限制分割后的最大单词数，若不填写，则默认全部分割。

  a. 字段样例

     Id：dc1dab7e-b045-4e77-bda4-914d083d1bf7

  b. 查询和分析语句

  ```
  SELECT split(Id,'-'), split(Id,'-',2)
  ```

  c. 查询和分析结果

  表 5-51 split 函数查询和分析结果

EXPR$0	EXPR$1
["dc1dab7e","b045","4e77","bda4","914d083d1bf7"]	["dc1dab7e","b045-4e77-bda4-914d083d1bf7"]

- **split_part函数**

  通过指定的分隔符拆分字符串，并返回指定索引的内容字段样例，索引下标从0开始。若索引下标超过分割数量或者为负数，则返回空字符串。

  a. 字段样例

     Id: dc1dab7e-b045-4e77-bda4-914d083d1bf7

  b. 查询和分析语句

  ```
  SELECT split_part(Id,'-',1)
  ```

  c. 查询和分析结果

表 5-52 split_part 函数查询和分析结果

EXPR$0
b045

- **split_to_map函数**

  用于使用指定的第一个分隔符拆分字符串，然后再使用指定的第二个分隔符进行第二次拆分，展示形式为{“KEY1”:”VALUE1”,”KEY2”:”VALUE2”}。无法被二次分割的value值为空。

  a. 字段样例

  Request:request_id:"e3ac4b70c7d244f080d434e300d8065a”;request_time: "1674965051000”

  b. 查询和分析语句

  ```
  SELECT split_to_map(Request,';',':')
  ```

  c. 查询和分析结果

  表 5-53 split_to_map 函数查询和分析结果

EXPR$0
{"request_id ":"e3ac4b70c7d244f080d434e300d8065a ","request_time":"1674965051000"}

5.7.11 SQL 比较运算符

功能描述

比较运算符用于比较两个值，并返回真(true)或假(false)。比较运算符可以对数值类型进行大小比较，对STRING类型进行包含比较，比如数值类型的字段num1 < num2是否为真，STRING类型的str1是否存在于字符串strs中等，具体请参见**表5-54**。

语法格式

```
SELECT fieldname1 WHERE fieldname1 > fieldname2
```

比较运算符语句

表 5-54 比较运算符语句

语句	说明	示例
`x = y`	等于。	SELECT num1 < num2
`x <> y`	不等于。	SELECT num1 <> num2
`x > y`	大于。	SELECT num1 > num2

语句	说明	示例
x >= y	大于或等于。	SELECT num1 >= num2
x < y	小于。	SELECT num1 < num2
x <= y	小于或等于。	SELECT num1 <= num2
x BETWEEN y AND z	等同于x >= y AND x <= z。	SELECT num1 BETWEEN num2 AND num3
x NOT BETWEEN y AND z	等同于x < y OR x > z。	SELECT num1 NOT BETWEEN num2 AND num3
x LIKE pattern	如果x匹配SQL LIKE模式，则返回true。	SELECT str1 LIKE '*'
x NOT LIKE pattern	如果x不匹配SQL LIKE模式，则返回true。	SELECT str1 NOT LIKE '*'
x IS NULL	如果x是空值或空白字符串，则返回true。	SELECT str1 IS NULL
x IS NOT NULL	如果x既不是空值也不是空白字符串，则返回true。	SELECT str1 IS NOT NULL
x IN (values)	如果x为其中一个列举值，则返回true。	SELECT str1 IN ('testStr1', 'testStr2')
x NOT IN (values)	如果x不在列举值中，则返回true。	SELECT str1 NOT IN ('testStr1', 'testStr2')
x IN (subquery)	如果x是通过指定子查询返回，则返回true。	SELECT str1 WHERE str2 IN (SELECT DISTINCT str2 LIMIT 100)
x NOT IN (subquery)	如果x不是通过指定子查询返回，则返回True。	SELECT str1 NOT IN (SELECT str2 LIMIT 100)

5.7.12 SQL IP 地址函数

功能函数

对于IPv4地址函数，地址参数可以是IPv4点分十进制字符串（例如"192.168.0.1"）或表示为整数的IP地址（例如3232235521）。subnet 参数应该是一个字符串，格式为CIDR表示法中的IPv4地址子网（例如"192.168.0.0/16"）

IP 函数语句

语句	说明	示例
IPV4_MATCH(address,subnet)	如果subnet属于address的子网地址则返回true，否则返回false。如果 address不是有效的IPv4地址，则返回false。如果address是整数而不是字符串，则此函数更效率。	SELECT IPV4_MATCH (address,subnet)
IPV4_PARSE(address)	将address解析为整数的IPv4地址。如果address是有效的IPv4地址，则它可以被解析。如果address不是有效的IPv4地址，则返回null。	SELECT IPV4_PARSE(address)
IPV4_STRINGIFY(address)	将整数address转换为以点分隔的IPv4地址字符串。如果address是有效的IPv4地址的整数，则它可以被解析。如果address不能表示为IPv4地址，则返回null。	SELECT IPV4_STRINGIFY(address)

示例及说明

IPV4_MATCH(address, subnet)函数

IPV4_MATCH函数，如果address属于subnet的子网ip，则返回true，否则返回false。如果address不是有效的IPv4地址，则返回false。如果address 是整数而不是字符串，则此函数具有更高的执行效率。

1. 字段样例

   Ipv4: 192.168.1.18

2. 查询和分析语句
```
select IPV4,IPV4_MATCH(Ipv4, '192.168.0.0/16')
```
3. 查询和分析结果

表 5-55 查询和分析结果

IPV4	EXPR$1
192.168.1.18	true

IPV4_PARSE(address)/ IPV4_STRINGIFY(address)函数

将address解析为整数的IPv4地址。如果address是有效的IPv4地址，则它可以被解析。如果address不是有效的IPv4地址，则返回null。

1. 字段样例

   Ipv4: 192.168.0.1

   Num: 3232235521
2. 查询和分析语句
```
select  IPV4_PARSE(Ipv4), IPV4_STRINGIFY(Num)
```
3. 查询和分析结果

表 5-56 查询和分析结果

EXPR$0	EXPR$1
-1062731775	192.168.0.1

5.7.13 SQL 归约函数

功能描述

归约函数对零个或多个表达式进行操作，并返回单个表达式。如果没有表达式作为参数传递，则结果为 NULL。表达式必须全部转换为公共数据类型，即结果的类型有：

- 如果所有的参数都是 NULL，结果是 NULL，否则，NULL 参数被忽略。
- 如果所有的参数包含了数字和字符串的混合，参数都被解释为字符串。
- 如果所有的参数是整型数字，参数都被解释为长整型。
- 如果所有的参数是数值且至少一个参数是double，则参数都被解释为double。

语法格式

```
GREATEST([expr1, ...])/ LEAST([expr1, ...])
```

示例及说明

GREATEST([expr1, ...])/ LEAST([expr1, ...])函数

GREATEST函数，计算零个或多个表达式，并根据上述比较返回最大值。

LEAST函数，计算零个或多个表达式，并根据上述比较返回最小值。

1. 字段样例

   Num: 11785730

2. 查询和分析语句

```
select  Num,GREATEST( "Num"/10,(select count(1)) ),LEAST("Num"/10,(select count(1)))
```

3. 查询和分析结果

表 5-57 归约函数查询和分析结果

Num	EXPR$1	EXPR$2
11785730	1178573	1

5.7.14 SQL 其他函数

功能描述

SQL提供的函数，还支持一些转换类型和CASE WHEN等逻辑运算，具体请参见**表 5-58**。

语法格式

```
SELECT CAST(fieldname1 AS VARCHAR) AS fieldname1_str
```

其他函数语句

表 5-58 其他函数语句

关键字	说明	示例
CAST(value AS TYPE)	转换数据类型。只支持转换为VARCHAR、FLOAT。	SELECT fieldname1, CAST(fieldname1 AS VARCHAR)
CASE WHEN boolean_expr1 THEN result1 \[WHEN boolean_expr2 THEN result2 ... \] \[ELSE resultN \] END	简单CASE函数。	SELECT CASE WHEN httpStatus = 200 THEN 1 ELSE 0 END
NULLIF(value1, value2)	如果value1和value2相等，返回空值，否则返回value1。	SELECT fieldname1, fieldname2, NULLIF(fieldname1, fieldname2)
NVL(expr,expr-for-null)	如果"expr"为空值或空白字符串，则返回"expr-for-null"。	SELECT NVL(str1, 'expr-for-null')

5.7.15 SQL JOIN 语法

JOIN子句可以关联查询两个或多个表数据，本文介绍JOIN子句的基本使用方法。

语法

```
select key
from t1
LEFT|RIGHT|INNER JOIN t2
on t1.key=t2.key
```

当前日志服务支持LEFT JOIN、RIGHT JOIN和INNER JOIN三种JOIN子句方式。具体功能如下：

表 5-59

JOIN方式	说明
LEFT JOIN	以左表（t1）的结果为基础，关联右表（t2）数据。 **说明** 当表名为纯数字时，需要给表名加上双引号转换成字符串。例如：表名是123，JOIN语句中该表应写成“123”。
RIGHT JOIN	以右表（t2）的结果为基础，关联左表（t1）数据。 **说明** 当表名为纯数字时，需要给表名加上双引号转换成字符串。例如：表名是123，JOIN语句中该表应写成“123”。
INNER JOIN	两个表的结果（elb1，elb2）交集数据

示例

有两个表，access表示主机的接入指标包含路径，时延，状态码，host为主机指标包含cpu和内存。通过JOIN可以关联接入和主机指标，查看相同主机的不同维度的指标情况。

- **LEFT JOIN**
  a. 查询语句
  ```
  SELECT
    "access".__time,
    "access".host_ip,
    "access".cost,
    "host".cpu,
    "host".memory
  FROM
    log "access"
    LEFT JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
  ```
  b. 返回结果，总共60条数据。
- **RIGHT JOIN**
  a. 查询语句
  ```
  SELECT
    "access".__time,
    "host".host_ip,
    "access".cost,
    "host".cpu,
    "host".memory
  FROM
    log "access"
    RIGHT  JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
  ```
  b. 返回结果，总共60条数据。

- **INNER JOIN**

  a. 查询语句

  ```
  SELECT
      "access".__time,
      "host".host_ip,
      "access".cost,
      "host".cpu,
      "host".memory
  FROM
      log "access"
      INNER  JOIN (select memory,cpu,host_ip from log) host ON "access".host_ip = "host".host_ip
  ```

  b. 返回结果，总共45条数据。

5.7.16 SQL 查询样例

本章以ELB日志为例进行介绍，对LTS中的ELB原始日志进行查询，具体查询步骤如下。

步骤1 登录云日志服务控制台，在日志管理页面，单击对应日志流名称，进入日志详情页面。

步骤2 系统获取ELB原始日志，在原始日志页面查看具体日志。

步骤3 单击右上角  ，在弹出页面中，选择“云端结构化解析”。

步骤4 选择结构化模板，根据ELB模板进行结构化配置。其他日志内容可选择其他结构化模板。

步骤5 在“可视化”页签下输入SQL查询语句对相应的字段进行查询，即可返回所需的日志内容。

----结束

查询结果呈现

表格

下面的语句查询请求的host，request_uri所对应的日志各有多少条，发送的请求体的大小（MB），请求返回的状态码分别是2xx, 3xx, 4xx, 5xx的占比，并按照日志条数降序排列。

```
SELECT "router_request_uri" as "request_uri", "host", COUNT(*) as pv,
round(sum(body_bytes_sent) / 1024.0 , 5) as "body_bytes_sent(MB)",
round(sum(case when status >= 200 and status < 300 then 1 else 0 end ) * 100.0 / COUNT(1), 6) as "2xx
ratio(%)",
round(sum(case when status >= 300 and status < 400 then 1 else 0 end ) * 100.0 / count(1), 6) as "3xx
ratio(%)",
round(sum(case when status >= 400 and status < 500 then 1 else 0 end ) * 100.0 / count(1), 6) as "4xx
ratio(%)",
round(sum(case when status >= 500 and status < 600 then 1 else 0 end ) * 100.0 / count(1), 6) as "5xx
ratio(%)"
GROUP BY "host", "router_request_uri"
ORDER BY pv DESC
LIMIT 100
```

柱状图

根据以下语句查询结果绘制柱状图，x轴选择"request_uri"，y轴选择"pv"，表示每种requestURL请求分别有多少条。

```
SELECT "router_request_uri" as "request_uri", "host", COUNT(*) as pv,
round(sum(body_bytes_sent) / 1024.0 , 5) as "body_bytes_sent(MB)",
```



```
round(sum(case when status >= 200 and status < 300 then 1 else 0 end ) * 100.0 / COUNT(1), 6) as "2xx
ratio(%)",
round(sum(case when status >= 300 and status < 400 then 1 else 0 end ) * 100.0 / count(1), 6) as "3xx
ratio(%)",
round(sum(case when status >= 400 and status < 500 then 1 else 0 end ) * 100.0 / count(1), 6) as "4xx
ratio(%)",
round(sum(case when status >= 500 and status < 600 then 1 else 0 end ) * 100.0 / count(1), 6) as "5xx
ratio(%)"
GROUP BY "host", "router_request_uri"
ORDER BY pv DESC
LIMIT 100
```

折线图

根据以下语句查询结果绘制折线图，x轴选择"_time_"，y轴选择"QPS"。表示查询时间段内间隔5s的QPS变化。

```
select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-
dd''T''HH:mm:ssZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_ , COUNT(*) as QPS from log group
by _time_
```

饼图

根据以下语句查询结果绘制饼图，类目选择"status"，数据选择"rm"，表示查询时间段内status不同值的占比。

```
SELECT status, COUNT(1) AS rm GROUP BY status
```

数字图

根据以下语句查询结果绘制数字图，查看最近一小时一共有多少次正常的请求。

```
SELECT count(*) AS normalRequest WHERE status = 200
```

6 日志搜索与分析（管道符方式-邀测）

6.1 日志搜索

云日志服务支持管道符特性在一个语句中同时进行搜索和分析。其语法结构主要由三部分构成：针对非结构化数据和半结构化数据的搜索语句、“|”和针对结构化数据的查询的分析语句。语法示例结构：* and msg:"hello world" | SELECT avg(value)

说明

目前此功能在邀测中，暂不支持申请开通。

目前支持两种查询方式（以kibana上的查询语句为例）：

- 情景1：仅包含sql查询
```
POST _opendistro/_sql/
{
"query": "select current_date() from employee"
}
```
- 情景2：包含lpl查询和sql查询
```
POST _opendistro/_sql/
{
"query": "age >22 | select name, count(name) from employee group by name"
}
```

用户如果仅需要使用lpl语句来查询，可以在管道符“|”的右侧补上select * from table.

数据组织形式

管道符特性针对特定的数据组织形式，对于云日志服务而言，管道符特性的数据主要由两部分构成，一部分是结构化数据，一部分是原始日志。示例如下：

{"id": 1, "name": "Bob", "job": "java", "age": 21, "sal": 8000, "gender": "female", "_log": "name: Bob, job: java, age: 21, sal: 8000, gender: female"}

其中：“id”，“name”，“job”，“age”，“sal”和“gender”是结构化数据的字段,“_log”是原始日志字段。

使用限制

- SQL函数不支持显式转为double、float类型，例如* | select cast(1 as **double**)
- SQL函数查询字段别名不支持包含.，比如* | select compare(`c.c`,1) as diff from (select count(status) as `**c.c**`) as t
- 索引字段名称中含有特殊字符的语句，需要使用半角符号“`”将字段名称包含起来。

日志搜索分析

步骤1 在云日志服务管理控制台，单击“日志管理”。

步骤2 在日志组列表中，单击日志组名称前对应的 按钮。

步骤3 在日志流列表中，单击日志流名称，进入日志详情页面。

步骤4 在“搜索分析”页签，在右上角选择时间范围，可以查看原始日志和统计图表。在搜索框由搜索语句和SQL分析语句组成，两者通过管道符|联动，详细请参考**SQL函数**。

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

说明

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤5 在日志内容下方，单击时间前面的 展开图标，支持以Table和JSON形式展示结构化字段。

- 在Table页签，支持对该字段进行添加到查询、从查询中排除、字段存在、字段不存在、隐藏的方式搜索日志。
- 在JSON页签，支持查看或者复制日志内容。

步骤6 版面设置。

1. 在下拉框单击编辑版面，进入版面设置页面，版面列表自带默认版面、纯净版面、容器日志默认版面，可以设置字段在版面是否显示。
   云端: 适用于有写权限的用户，版面配置信息保存在云端。
   本地缓存: 适用于只有读权限的用户，版面配置信息缓存在本地浏览器。
2. 单击 新增自定义版面，设置版面名称和版面字段的可见性。
3. 设置完成后，单击“确定”，返回下拉框显示新增的自定义版面。

步骤7 交互式搜索分析，适用于生成简单的分析语句，如果您需要使用更多的函数或者嵌套查询，请手动输入SQL语句。

1. 单击搜索框前面的 按钮，进入交互式搜索分析页面。
2. 选择日志搜索的内容和条件。支持添加关联关系或添加组。

3. 设置完成后，支持预览搜索语句。
4. 在日志分析下方，进行添加指标和分组排序。通过统计函数计算出想要的指标信息，然后基于选定的字段进行分组排序。
5. 设置完成后，单击“确定”，即可在搜索框快速生成分析语句。

----结束

日志搜索的常用操作

日志搜索的常用操作有分享日志、刷新等操作，具体参考如下图所示：

表 6-1 常用操作

操作	说明
创建快速查询	单击 按钮，创建快速查询。
查看仪表盘	单击 按钮，在弹出来的仪表盘页面，可查看已创建的仪表盘。
添加告警	单击 按钮，在弹出的页面，支持**新建告警规则**。
分享日志	单击 复制当前日志搜索页面的链接，用于分享搜索日志。
刷新日志	单击 对日志进行刷新，有两种方式刷新方式：手动刷新和自动刷新。 ● 手动刷新：单击“手动刷新”可直接对日志进行刷新。 ● 自动刷新：选择自动刷新的间隔时间，将对日志进行自动刷新。间隔时间范围为15秒、30秒、1分钟和5分钟。
复制	单击 复制日志内容。
查看上下文	单击 查看日志上下文。 **说明** 支持选择简洁模式查看日志上下文。支持下载上下文内容。
简化字段详情	单击 查看简化字段详情。
换行/取消换行	单击 按钮，搜索的日志内容将换行显示。若不需要换行，单击 按钮，取消换行。 **说明** 默认开启换行按钮。

操作	说明
下载日志	单击  按钮，在弹出的下载日志页面中单击“本地下载”和“前往创建转储”。 本地下载：将日志文件直接下载到本地，单次下载支持最大5,000条日志。 在下拉框中选择“.csv”或“.txt”，单击“开始下载日志”，可将日志导出至本地。 **说明** • 选择以CSV格式导出日志后，本地以表格形式保存日志的具体标签信息。 • 选择导出TXT格式日志后，本地会以.txt格式保存日志的日志内容。 前往创建转储：通过OBS转储任务下载日志文件，单次下载支持最大100万条日志。 单击“前往创建转储”，跳转至配置转储页面，详细请参考**日志转储**。
全部折叠/全部展开	单击  设置日志内容展示的行数。若不需要展示日志内容，再单击一次  按钮即可关闭展示的日志内容。 **说明** 默认不折叠。折叠后，默认显示2行，最多支持展示6行。
JSON设置	鼠标悬浮在  按钮上，单击“JSON设置”，在弹出的JSON设置页面中，设置格式化显示。 **说明** 默认开启格式化，JSON默认展开层级为2层。若日志包含多个反斜杠，当日志展示为json格式时，会丢失一个反斜杠，因为json解析会将第一个反斜杠作为转义符处理。 • 开启格式化按钮：设置JSON默认展开层级，最大设置为10层。 • 关闭格式化按钮：对于JSON格式的日志，将不会格式化层级显示。
日志折叠设置	鼠标悬浮在  按钮上，单击“日志折叠设置”，在弹出的日志折叠设置页面中，设置长日志字符个数。 日志超过设置的长日志字符个数时，超出字符将被隐藏，单击“展开”按钮可查看全部内容。 **说明** 默认开启自动折叠长日志，字符个数默认为400个。
日志时间展示	鼠标悬浮在  按钮上，单击“日志时间展示”，在弹出的日志折叠设置页面中，设置是否展示毫秒、是否展示时区。 **说明** 默认开启展示毫秒。

操作	说明
虚拟滚动设置	鼠标悬浮在  按钮上，单击“虚拟滚动设置”，在弹出的虚拟滚动设置页面中，设置是否开启虚拟滚动、填写缓冲区大小。 **说明** • 虚拟滚动可以避免或减少滚动时卡顿的情况，提升操作体验，防止页面卡死。 • 滚动时数据会重新渲染，一定程度上影响数据流畅性。 • 缓冲区决定同时加载的数据量大小，缓冲区越大，同时加载的数据越多，但滚动性能会越差。
不可见字段列表 	该列表展示版面设置中配置的不可见性字段。 • 当日志流未配置版面设置时，将不显示  按钮。 • 当日志内容为“CONFIG_FILE”且未配置版面设置时，不可见字段默认有appName、clusterId、clusterName、containerName、hostIPv6、NameSpace、podName和serviceID。

6.2 设置 LTS 日志索引配置

索引是一种存储结构，用于对日志数据进行查询。通过配置索引后，可对日志进行查询和分析操作。不同的索引配置，则会产生不同的查询和分析结果，请根据您的需要，合理配置索引。

日志示例

以下是一条典型日志，content字段值是日志原文，使用分隔符逗号将原始日志解析成3个字段level、status、message；

示例日志中的hostName、hostIP、pathFile是常见的内置保留字段，详细内置字段请参考**内置保留字段**。

```
{
"hostName":"epstest-xx518",
"hostIP":"192.168.0.31",
"pathFile":"stdout.log",
"content":"error,400,I Know XX",
"level":"error",
"status":400,
"message":"I Know XX"
}
```

索引类型

云日志服务的索引类型如下：

表 6-2 索引类型表

索引类型	说明
全文索引	开启全文索引后，日志服务根据您设置的分词符将整条日志所有字段值拆分成多个词并构建索引。 **说明** • 用户上传的自定义标签（label）字段，不包含在全文索引中，如果您需要搜索自定义标签字段，请添加对应的字段索引。 • LTS内置保留字段，不包含在全文索引中，您需要通过字段索引Key:Value的方式进行搜索，请参考**内置保留字段**。
字段索引	配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询，缩小查询范围。 **说明** • 日志服务默认为部分内置保留字段创建字段索引，请参考**内置保留字段**。 • 如果您的某个字段单独配置了字段索引，那么该字段值的分词符以字段索引配置为准。 • 结构化配置中的的快速分析列已被移除，如果您要使用**快速分析功能**，则必须配置字段索引且开启对应字段的快速分析按钮。 关于日志示例有两种情况： • 在日志示例中，配置了level和status两个字段索引，其中level是string类型，字段值是error，单独配置了分词符，status是long类型，不需要配置分词符；您可以使用level : error的方式精确搜索level字段值为error的所有日志。 • 在日志示例中，云日志服务LTS会默认为hostName、hostIP、pathFile这些内置保留字段创建字段索引。

注意事项

- 全文索引属性和字段索引属性必须至少启用一种。
- 创建索引会产生索引流量和索引存储空间，费用说明请参见**价格计算器**。
- 索引配置（新增、编辑、删除字段，修改配置项等操作）只对新写入的日志生效，历史日志不会生效。当前不支持对历史日志重建索引。
- 关闭索引后，历史索引的存储空间将在当前日志流的数据保存时间到期后，自动被清除。
- 日志服务默认已为部分内置保留字段创建字段索引，请参见**内置保留字段**。
- 不同的索引配置，会产生不同的查询和分析结果，请根据您的需求，合理创建索引。全文索引和字段索引互不影响。
- 索引配置修改后，对新写入的日志数据生效，历史日志数据不会生效。
- 在字段索引功能上线前，SQL分析支持的字段来自于云端结构化解析；在字段索引功能上线后，只要用户配置了字段索引，SQL分析支持的字段将来自于字段索引，因此修改字段索引可能对现有的可视化图表、仪表盘、SQL告警、定时SQL、Grafana接入中的查询结果产生影响，请谨慎操作！
- 字段索引配置为JSON数据类型时，在原始日志页面不支持对JSON子字段使用快速分析、跳转图表，不支持可视化；在搜索分析页面下支持JSON子字段使用快速分析和SQL可视化功能。
- 字段索引配置为JSON数据类型时，对JSON父字段查询时支持对命中结果高亮，对JSON子字段查询时不支持对命中结果高亮。

索引流量

创建索引后，会产生索引流量，索引流量的详细计算规则请参考**价格计算器**。

内置保留字段

在采集日志时，日志服务会将采集时间、日志类型、主机IP等信息以Key-Value对的形式添加到日志中，这些字段是云日志服务的内置字段。

说明

- 使用API写入日志数据或添加ICAgent配置时，请不要将字段名称设置为内置保留字段，否则可能会造成字段名称重复、查询不精确等问题。
- 日志服务为日志数据增加的内置保留字段当前免费，后续会按照按量付费方式正常收费（为其开启索引时也会产生少量索引流量及存储费用）。更多信息请参见**价格计算器**。
- 用户自定义日志字段名称中不能使用双下划线__，否则无法配置索引。

表 6-3 内置保留字段说明

内置保留字段	数据格式	索引与统计设置	说明
collectTime	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为collectTime创建字段索引，索引数据类型为long类型。 查询时输入 collectTime : xxx。	采集时间，指日志被采集器ICAgent采集时的时间。 示例中的"collectTime":"1681896081334"，转换成标准时间是2023-04-19 17:21:21
__time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为time创建字段索引，索引数据类型为long类型。该字段不支持查询。	日志时间，指的是日志在控制台页面展示的日志时间。 例如示例中的"__time__":"1681896081334"，转换成标准时间是2023-04-19 17:21:21 日志时间默认使用采集时间，也支持自定义日志时间。

内置保留字段	数据格式	索引与统计设置	说明
lineNum	整型	索引设置：开启索引后，日志服务默认为lineNum创建字段索引，索引数据类型为long类型。	行号（偏移量），用来排序日志。 非高精度日志会根据collectTime生成，默认是collectTime * 1000000 + 1，高精度日志就是用户上报的纳秒值。 例如示例中的"lineNum":"1681896081333991900"。
category	字符串	索引设置：开启索引后，日志服务默认为category创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入category: xxx。	日志类型，表示该日志的来源。 例如ICAgent采集的日志该字段为LTS，某云服务例如DCS上报的日志该字段为DCS。
clusterName	字符串	索引设置：开启索引后，日志服务默认为clusterName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterName: xxx。	集群名称，k8s场景下集群名称。 例如示例中的"clusterName":"epstest"。
clusterId	字符串	索引设置：开启索引后，日志服务默认为clusterId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入clusterId: xxx。	集群ID，k8s场景下集群ID。 例如示例中的"clusterId":"c7f3f4a5-xxxx-11ed-a4ec-0255ac100b07"。
nameSpace	字符串	索引设置：开启索引后，日志服务默认为nameSpace创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入nameSpace: xxx。	命名空间，k8s场景下命名空间。 例如示例中的"nameSpace":"monitoring"。

内置保留字段	数据格式	索引与统计设置	说明
appName	字符串	索引设置：开启索引后，日志服务默认为appName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入appName: xxx。	组件名称，k8s场景下工作负载的名称。 例如示例中的"appName":"alertmanager-alertmanager"。
serviceID	字符串	索引设置：开启索引后，日志服务默认为serviceID创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入serviceID: xxx。	工作负载ID，k8s场景下工作负载ID。 例如示例中的"serviceID":"cf5b453xxxad61d4c483b50da3fad5ad"。
podName	字符串	索引设置：开启索引后，日志服务默认为podName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podName: xxx。	POD名称，k8s场景下POD名称。 例如示例中的"podName":"alertmanager-alertmanager-0"。
podIp	字符串	索引设置：开启索引后，日志服务默认为podIp创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入podIp: xxx。	pod的ip，k8s场景下pod的IP地址。 例如示例中的"podIp":"10.0.0.145"。
containerName	字符串	索引设置：开启索引后，日志服务默认为containerName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入containerName: xxx。	容器名称，k8s场景下容器名称。 例如示例中的"containerName":"config-reloader"。
hostName	字符串	索引设置：开启索引后，日志服务默认为hostName创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostName: xxx。	主机名称，ICAgent所在主机的名称。 例如示例中的"hostName":"epstest-xx518"。

内置保留字段	数据格式	索引与统计设置	说明
hostId	字符串	索引设置：开启索引后，日志服务默认为hostId创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostId: xxx。	主机ID，ICAgent所在主机的id，该id由ICAgent生成。 例如示例中的"hostId":"318c02fe-xxxx-4c91-b5bb-6923513b6c34"。
hostIP	字符串	索引设置：开启索引后，日志服务默认为hostIP创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIP: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv4场景） 例如示例中的"hostIP":"192.168.0.31"。
hostIPv6	字符串	索引设置：开启索引后，日志服务默认为hostIPv6创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入hostIPv6: xxx。	主机IP，日志采集器所在主机的ip（适用于ipv6场景） 例如示例中的"hostIPv6":""。
pathFile	字符串	索引设置：开启索引后，日志服务默认为pathFile创建字段索引，索引数据类型为string类型，分词字符为空。查询时输入pathFile: xxx。	文件路径，采集的日志文件的路径。 例如示例中的"pathFile":"stdout.log"。
content	字符串	索引设置：开启全文索引后，会使用全文索引定义的分词符对content字段的value进行分词；不支持将content字段配置到字段索引中。	日志原文 例如示例中的"content":"level=error ts=2023-04-19T09:21:21.333895559Z"
__receive_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__receive_time__创建字段索引，索引数据类型为long类型。	上报日志的服务端时间，相当于LTS采集端接收到日志的时间。

内置保留字段	数据格式	索引与统计设置	说明
__client_time__	整型，Unix时间戳（毫秒）	索引设置：开启索引后，日志服务默认为__client_time__创建字段索引，索引数据类型为long类型。	端侧日志的客户端上报时间。
_content_parse_fail_	字符串	索引设置：开启索引后，日志服务默认为_content_parse_fail_创建字段索引，索引数据类型为string类型，分词字符为默认分词符。查询时输入_content_parse_fail_:xxx。	上报日志解析失败的日志内容。
__save_time__	整型，Unix时间戳（毫秒）	不支持将__save_time__字段配置到字段索引中。	日志流引擎的时间字段，根据此时间拉取对应时间段内的日志数据。
__time	整型，Unix时间戳（毫秒）	不支持将__time字段配置到字段索引中。	不涉及。
logContent	字符串	不支持将logContent字段配置到字段索引中。	不涉及。
logContentSize	整型	不支持将logContentSize字段配置到字段索引中。	不涉及。
logIndexSize	整型	不支持将logIndexSize字段配置到字段索引中。	不涉及。
groupName	字符串	不支持将groupName字段配置到字段索引中。	不涉及。
logStream	字符串	不支持将logStream字段配置到字段索引中。	不涉及。

配置全文索引

步骤1 登录云服务日志控制台，单击“日志管理”。

步骤2 在日志组列表中，单击日志组名称左侧的 ，选择日志流，进入日志流管理界面。

步骤3 在日志流详情页面，单击右上角 ，进入索引配置页面。

步骤4 在索引配置页面中，默认开启“全文索引”按钮。

说明

- 在索引配置页面选择自动配置时，默认获取最近15分钟的原始日志和内置字段的交集，LTS自动将原始日志和内置字段的交集、当前结构化字段、tag字段一起组成字段索引下方的表格数据。
- 若15分钟内没有原始日志，则获取hostIP、hostName、pathFile、结构化字段、tag字段结合共同组成字段索引下方的表格数据。
- ECS接入选择结构化配置时，进入索引配置页面，则会自动加上如下字段：category、hostName、hostId、 hostIP、 hostIPv6、 pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。
- CCE接入选择结构化配置时，进入索引配置页面，则会自动加上如下字段：category、clusterId、 clusterName、 nameSpace、 podName、 containerName、 appName、hostName、 hostId、 hostIP、 hostIPv6、 pathFile，添加字段时，若某个字段已存在于索引配置，则不会重复添加。

步骤5 请参考表6-4配置参数信息。

表 6-4 自定义全文索引配置参数

参数	说明
全文索引	打开全文索引开关，表示创建全文索引。
大小写敏感	查询时是否区分英文字母的大小写。 • 打开大小写敏感开关，则查询时区分大小写。例如示例日志含有Know，那么您只能使用**Know**才能查询到该日志。 • 关闭大小写敏感开关，则查询时不区分大小写。例如示例日志含有Know，那么您使用关键字**KNOW**和**know**都能查到该日志。

参数	说明
包含中文	查询时是否区分中英文。 ● 打开包含中文开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 **说明** – 一元分词是指将中文字符串拆分为一个个独立的中文字。 – 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 – 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 ● 关闭包含中文开关后，按照分词符的设置拆分所有内容。 例如示例日志内容为： **error,400,I Know 今天是星期一**。 ● 关闭包含中文开关后，按照分词符的设置拆分英文内容，日志会被拆分为**error**、**400**、**I**、**Know**、**今天是星期一**，您可以通过**error**或**今天是星期一**查找该日志。 ● 打开包含中文开关后，日志服务后台分词器将日志拆分为**error**、**400**、**I**、**Know**、**今**、**天**、**是**、**星**、**期**、**一**，您通过**error**或**今天**等词都可以查找到该日志。
分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。所有的ASCII码包括中文都可被定义为分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 例如示例日志内容为： **error,400,I Know 今天是星期一**。 ● 如果不设置任何分词符，整条日志被作为一个词**error,400,I Know 今天是星期一**，您只能通过完整字符串**error,400,I Know 今天是星期一**或模糊查询**error,400,I K***查找该日志。 ● 如果设置分词符为逗号（,），则原始日志被拆分为**error**、**400**、**I Know 今天是星期一**3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**error**、**400**、**Kn***、**今天是***。 ● 如果设置分词符为逗号（,）和空格，则原始日志被拆分为**error**、**400**、**I**、**Know**、**今天是星期一**5个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**Know**、**今天是***。
特殊分词符	单击“添加特殊分词符”，参考**ASCII码对照表**输入ASCII值。

步骤6 完成后，单击确定。

----结束

配置字段索引

创建字段索引时，最多支持添加500个字段。其中JSON类型字段，最多支持添加100个子字段。

说明

字段索引的自定义分词符和特殊分词符仅支持白名单用户提交工单申请使用。详细操作请参考**提交工单**。

步骤1 配置全文索引后，在索引配置页面的日志分析下方，单击开启可视化后，配置的字段索引支持SQL可视化分析，否则无法查询到ICAgent结构化的可视化数据。

步骤2 在索引配置页面的字段索引下方，单击“添加字段”，参考**表6-5**设置字段信息。

步骤3 或者勾选字段，单击批量配置，在批量配置页面设置参数。

图 6-1 批量配置

步骤4 参考**表6-5**配置字段索引。

说明

- 字段索引的参数配置仅对该字段生效。
- 当添加的字段在日志内容中不存在时，则配置的该索引字段无效。

表 6-5 自定义字段索引配置参数

参数	说明
字段名称	日志字段名称，例如示例日志中的level。 字段名称只能包括字母、数字或下划线（_），且只能以字母或下划线（_）开头，字段名称中不能含有双下划线。 **说明** • 双下划线（__）在LTS不对用户呈现的内置保留字段中使用，用户自定义日志字段名中不能使用双下划线__，否则无法配置字段索引名称。 • 日志服务默认会对部分内置保留字段开启字段索引，请参见**内置保留字段**。 • 若是内置字段，在字段名称后会显示“内置”字眼，方便用户识别。
执行操作	显示字段的添加状态：新增、不修改、修改、删除。索引字段有变动后，单击“修改对比”，即可查看原配置内容与修改后配置内容的差异。 • 显示新增的字段不支持修改执行操作。 • 修改类型、大小写敏感、自定义分词符、特殊分词符、包含中文、快速分析时，会与原索引配置中的字段进行对比，若任意一项不同，则执行操作变为“修改”。 • 索引配置单击确定后，不会保存执行操作为“删除”的字段。
类型	• 日志字段值（Value）的数据类型，可选值为string、long、float、json。 **说明** 字段json类型只对ICAgent结构化解析生效，对云端结构化解析不生效。 • long类型和float类型不支持设置大小写敏感、包含中文和分词符。
大小写敏感	查询时是否区分英文字母的大小写。 • 打开大小写敏感开关，则查询时区分大小写。例如示例日志message字段中含有Know，那么您只能使用**message:Know**才能查询到该日志。 • 关闭大小写敏感开关，则查询时不区分大小写。例如示例日志message字段中含有Know，那么您使用关键字**message:KNOW**和**message:know**都能查到该日志。

参数	说明
自定义分词符	根据指定分词符，将日志内容拆分成多个词。当默认设置不能满足您的需求时，您可以自定义设置分词符。所有的ASCII码包括中文都可被定义为分词符。 如果设置分词符为空，则字段值将被当成一个整体，您只能通过完整字符串或模糊查询查找对应的日志。 例如示例日志message字段内容为：**I Know 今天是星期一**。 • 如果不设置任何分词符，整条日志被作为一个词**I Know 今天是星期一**，您只能通过完整字符串**message:I Know 今天是星期一**或模糊查询**message:I Know 今天是*查找该日志**。 • 如果设置分词符为空格，则原始日志被拆分为**I**、**Know**、**今天是星期一**3个词，您通过任意一个词或词的模糊查询都可以找到该日志，例如**message:Know**或**message:今天是星期一**。
特殊分词符	单击“添加特殊分词符”，参考**ASCII码对照表**输入ASCII值。
包含中文	查询时是否区分中英文。 • 打开包含中文开关后，如果日志中包含中文，默认按照一元分词法拆分中文内容，按照分词符的设置拆分英文内容。 **说明** – 一元分词是指将中文字符串拆分为一个个独立的中文字。 – 使用一元分词符的优点是对海量日志分词效率高，其他中文分词方法对写入速度影响大。 – 打开包含中文功能，会对中文使用一元分词（每个汉字单独分词），如果需要更精确的搜索结果，请用短语搜索，语法为：#"待搜索的短语"。 • 关闭包含中文开关后，按照分词符的设置拆分所有内容。 例如示例日志message字段内容为：**I Know 今天是星期一**。 • 关闭包含中文开关后，按照分词符的设置拆分英文内容，日志会被拆分为**I**、**Know**、**今天是星期一**，您可以通过**message:Know**或**message:今天是星期一**查找该日志。 • 打开包含中文开关后，日志服务后台分词器将日志拆分为**I**、**Know**、**今**、**天**、**是**、**星**、**期**、**一**，您通过**message:Know**或**message:今天**等词都可以查找到该日志。
快速分析	默认为开启状态，开启后，可以对字段值做采样统计，请参见**11.6.4-快速分析**。 **说明** • 快速分析的原理是对搜索命中的日志采样10万条进行数据统计，不是全量统计。 • 快速分析的字段长度最大为2000字节。 • 快速分析字段展示前100条数据。
操作	单击  ，删除添加的自定义字段。

步骤5 完成后，单击“确定”。

----结束

自动配置字段索引

在创建字段索引时，您可以单击自动配置，日志服务会自动添加一些字段索引，您可以根据自己的需要增加或者删除字段：

- 日志服务会根据采集时预览数据中的第一条内容，自动生成字段索引。
- 日志服务会选取几个最常见的内置保留字段添加到字段索引中（例如hostIP、hostName、pathFile）。

ASCII 码对照表

表 6-6 ASCII 码对照表

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL（空字符）	32	空格	64	@	96	`
1	SOH（标题开始）	33	!	65	A	97	a
2	STX（正文开始）	34	"	66	B	98	b
3	ETX（正文结束）	35	#	67	C	99	c
4	EOT（传输结束）	36	$	68	D	100	d
5	ENQ（询问字符）	37	%	69	E	101	e
6	ACK（确认回应）	38	&	70	F	102	f
7	BEL（响铃）	39	'	71	G	103	g
8	BS（退格）	40	(	72	H	104	h
9	HT（水平定位符号，制表符）	41	)	73	I	105	i
10	LF（换行）	42	*	74	J	106	j
11	VT（垂直定位符号）	43	+	75	K	107	k
12	FF（换页键）	44	,	76	L	108	l

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
13	CR（归位键）	45	-	77	M	109	m
14	SO（取消变换）	46	.	78	N	110	n
15	SI（启用变换）	47	/	79	O	111	o
16	DLE（跳出数据通讯）	48	0	80	P	112	p
17	DC1（设备控制1）	49	1	81	Q	113	q
18	DC2（设备控制2）	50	2	82	R	114	r
19	DC3（设备控制3）	51	3	83	S	115	s
20	DC4（设备控制4）	52	4	84	T	116	t
21	NAK（确认失败回应）	53	5	85	U	117	u
22	SYN（同步用暂停）	54	6	86	V	118	v
23	ETB（区块传输结束）	55	7	87	W	119	w
24	CAN（取消）	56	8	88	X	120	x
25	EM（连接介质中断）	57	9	89	Y	121	y
26	SUB（替换）	58	:	90	Z	122	z
27	ESC（跳出）	59	;	91	[	123	{
28	FS（文件分割符）	60	<	92	\	124	|
29	GS（组群分隔符）	61	=	93	]	125	}
30	RS（记录分隔符）	62	>	94	^	126	~
31	US（单元分隔符）	63	?	95	_	127	DEL（删除）

6.3 云端结构化解析

6.3.1 日志结构化概述

日志数据可分为结构化数据和非结构化数据。结构化数据指能够用数字或统一的数据模型加以描述的数据，具有严格的长度和格式。非结构化数据指不便于用数据库二维逻辑表来表现的数据，数据结构不规则或不完整，没有预定义的数据模型。

日志结构化是以日志流为单位，通过不同的日志提取方式将日志流中的日志进行结构化，提取出有固定格式或者相似程度较高的日志，过滤掉不相关的日志，以便对结构化后的日志按照SQL语法进行查询与分析。

注意事项

- 日志结构化是以日志流为单位，请先创建一个日志流。
- 日志流中的大部分日志需有一定的规则，否则结构化是无意义的。
- 结构化配置修改后，对新写入的日志数据生效，历史日志数据不会生效。

创建结构化配置

通过对日志流添加提取规则将日志流中的原始日志按一定的规律进行提取，并将提取后的日志整合到一起，以便进行SQL查询与分析。

下面详细介绍原始日志结构化的操作步骤：

步骤1 登录LTS控制台，在左侧导航栏中选择“日志管理”。

步骤2 结构化日志以日志流为单位，请在“日志管理”页面选择目标日志组和日志流。

步骤3 在日志流详情页面，单击右上角  ，在弹出页面中，选择“云端结构化解析”，进入日志结构化配置页面，选择对应的日志提取方法进行配置。

- **正则分析**
- **JSON**
- **分隔符**
- **Nginx**
- **结构化模板**

结构化后的日志数据可理解为数据库中的二维表，接下来就可以使用SQL语句对提取的字段进行查询与分析。

说明

- 开启“自动配置索引和快速分析”开关，将使用结构化字段配置字段索引，同时会打开快速分析按钮。开启该按钮后，默认将内置字段hostIP、hostName和pathFile配置为索引字段，并打开快速分析。
- 如果结构化后的字段长度超过20k字节时，仅会保留前20k字节长度。
- 结构化不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime、category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。

步骤4 开启**自定义日志时间**。

步骤5 完成后，单击“保存”。

----结束

修改结构化配置

结构化配置创建完成后，如果您需要修改结构化配置时，操作步骤如下：

步骤1 在结构化配置页面中，单击 ，可修改结构化配置。

说明

- 修改结构化配置支持修改结构化方式、日志提取字段和tag字段等。
- 系统模板不支持修改。

步骤2 完成后，单击“保存”。

----结束

删除结构化配置

如果日志结构化配置不再使用，可以删除结构化配置，操作步骤如下：

步骤1 在结构化配置页面中，单击 ，可删除结构化配置。

步骤2 在弹出对话框中，单击“确定”。

说明

删除结构化配置后，无法恢复，请谨慎操作。

----结束

6.3.2 设置日志云端结构化解析

云日志服务（LTS）目前支持5种日志结构化方式，分别是正则分析、JSON、分隔符、Nginx和结构化模板。您可以根据日志内容的实际场景进行选择。

使用限制

结构化字段最大长度为16KB，超过部分会被截断。

正则分析

正则分析是使用正则表达式提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。

- **从已有日志中选择**：单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，单击“确定”。通过选择不同时间段筛选日志。
- **从剪切板中粘贴**：单击“从剪切板中粘贴”，可直接自动将您剪切的日志内容复制到示例日志框中。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 字段提取。包括自动生成和手动输入两种方式，可将选择的日志提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

- 自动生成：当用户选择自动生成时，可以用鼠标选中示例日志中待结构化的日志内容，在弹出的对话框中为选中内容设置一个名称，名称必须以字母开始，且仅包含字母和数字，单击“添加”。
- 手动输入：当用户选择手动输入时，可以在输入框中输入正则表达式，单击“生成字段”来进行字段提取。正则表达式通过分组来捕获字段，分组指用圆括号"()"括起来的正则表达式，匹配出的内容就表示一个分组，分组包含如下三种形式：
  - (exp)：把括号内的正则作为一个分组，系统自动分配组号，规则为从正则表达式的左边开始，第一个左括号”(”对应第一个分组，第二个”(”对应第二个分组，依次类推，组号从1开始，从左向右，依次累加。
  - (?<name>exp)：表示命名分组，分组的正则表达式为exp，分组名为name。分组名必须以字母开始，且仅包含字母和数字，可以通过分组名或分组号引用该分组。
  - (?:exp)：表示不捕获分组，该分组只在当前位置匹配文本，在该分组之后，无法引用该分组，因为该分组没有分组名，没有分组号，也不会占用分组编号。

说明

- 分词符指将日志内容切分为多个单词的符号，默认分词符包括, '";=()[]{}@&<>/:\\?\n\t\r，在日志搜索或者对日志进行结构化时，可以选取相邻两分词符之间的单词。
- 在手工输入方式中，正则表达式的长度不能超过5000个字符，不强制要求用户在输入正则表达式时对分组进行命名，单击“生成字段”会以命名分组中的分组名作为字段名称，对于非命名分组会提取出对应的字段，并给字段名称默认命名field1、field2、field3……。

步骤3 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

JSON

JSON是通过提取JSON字段将其拆分为键值对。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤2 字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

将以下原始日志输入待操作框中。

```
{"a1": "a1", "b1": "b1", "c1": "c1", "d1": "d1"}
```

说明

- 当日志提取字段的类型为float时，精度为16位有效数字。如果超过16位有效数字，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。
- 当日志提取字段的类型为long时，日志内容超过16位有效数字，只会精确显示前16位有效数字，后面的数字会变为0。
- 当日志提取字段的类型为long时，日志内容超过21位有效数字，则会识别为float类型，建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤3 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

分隔符

分隔符是使用分隔符（例如：逗号、空格或字符）提取字段。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 在“步骤2 指定分隔符”需要根据原始日志内容选择分隔符，或自定义其他需要的特殊字符作为分隔符。

说明

- 不可见字符需要输入0x开头的16进制字符，长度为0-4个字符，总共32个不可见字符。
- 自定义字符支持输入1-10个字符，每个字符都作为独立的分隔符。
- 自定义字符串支持输入1-30个字符，字符串整体作为一个分隔符。

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤3字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

将以下原始日志输入待操作框中。

```
1 5f67944957444bd6bb4fe3b367de8f3d 1d515d18-1b36-47dc-a983-bd6512aed4bd 192.168.0.154 192.168.3.25 38929 53 17 1 96 1548752136 1548752736 ACCEPT OK
```

说明

当日志提取字段的类型为float时，精确度为7位有效数字。

如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

Nginx

Nginx是通过log_format指令来自定义访问日志的格式。

步骤1 选择示例日志：应选择一条比较典型的日志作为示例日志。在“步骤1 选择示例日志”中，可单击“从已有日志中选择”，在弹出框中根据业务需求选择待操作的日志，也可以直接在输入框中输入待操作的日志，单击“确定”。通过选择不同时间段筛选日志。

说明

时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据

步骤2 在“步骤2 输入Nginx日志配置”中需要输入Nginx日志配置，根据输入或选择的日志进行配置。其中有默认配置可使用，单击“默认Nginx配置”即可。

说明

标准Nginx配置文件中，日志配置的部分通常以log_format开头。

日志格式

- 默认配置如下所示。

```
log_format  main   '$remote_addr - $remote_user [$time_local] "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';
```

- 用户也可进行自定义配置，具体配置格式要求如下所示。
  - 使用Nginx配置，不可为空
  - 以log_format开头，并且包含（'）和字段名称
  - 长度最大限制为5000
  - 需要与示例日志内容匹配
  - log_format字段之间的间隔，除大小字母、数字、下划线及中划线外，可使用其他任意字符
  - 以（'）或者（';）结尾

步骤3 字段提取。可将输入或选择的日志自动提取为以一个示例字段对应一个字段名称的格式的日志解析结果。

在“步骤3 字段提取”下单击“智能提取”。以如下原始日志为例进行分析：

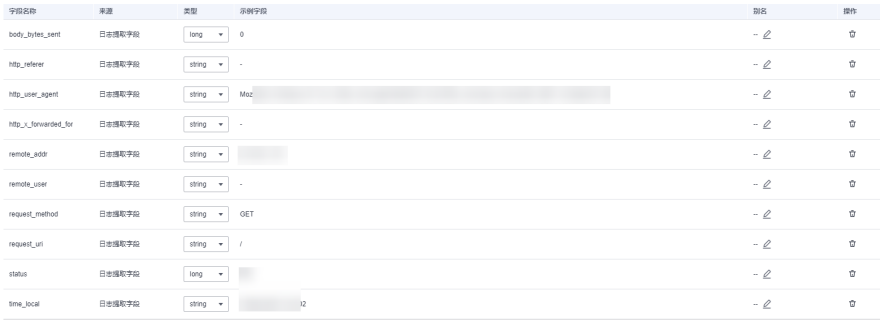
将以下原始日志输入待操作框中。

```
39.149.31.187 - - [12/Mar/2020:12:24:02 +0800] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.132 Safari/537.36" "-"
```

并使用如下Nginx日志配置。

```
log_format  main   '$remote_addr - $remote_user [$time_local] "$request" '
                   '$status $body_bytes_sent "$http_referer" '
                   '"$http_user_agent" "$http_x_forwarded_for"';
```

图 6-2 智能提取结果



说明

- 当日志提取字段的类型为float时，精确度为7位有效数字。
- 如果超过7位有效数字的话，则会导致提取字段内容不准确，从而影响可视化查看和快速分析，因此建议将字段类型修改为String。

在字段提取完成后，可对日志模板进行设置。结构化字段设置规则请参考**设置结构化字段**。

步骤4 单击“保存”，完成日志结构化配置，初次设置完成后将不能对字段类型编辑修改。

----结束

结构化模板

结构化模板是通过自定义模板或系统内置模板提取字段。

详情请参考**结构化模板**。

6.3.3 设置云端结构化模板

云日志服务（LTS）目前支持的结构化模板有两种：系统模板和自定义模板。

系统模板

支持多种系统模板，不支持修改系统模板的字段类型和删除字段，详情请参考**表6-7**。

步骤1 在“选择模板”下，选择“系统模板”，选择对应的系统模板，模板日志从对应的云服务接入，可以直接应用模板的数据模型作为示例日志。

步骤2 选择模板后“模板详情”中会自动显示对应的日志解析结果。单击“保存”完成结构化配置。

说明

- 结构化配置时，如果使用系统模板，则系统模板中的时间为自定义日志时间。支持通过模板名称搜索模板，方便用户快速查询模板信息。
- string类型的字段不支持使用运算符（>=<）或 in 语法进行范围查询，建议使用星号（*）或问号（?）进行模糊查询。需要重新配置结构化，将该字段修改为数字类型。

表 6-7 系统模板

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
ELB模板	根据ELB资料中提供的日志字段被定义。	否	否	**弹性负载均衡 ELB接入LTS**
VPC模板	根据VPC资料中提供的日志字段被定义。	否	否	**虚拟私有云VPC接入LTS**
CTS模板	字段名称为json日志中的key。	否	否	**云审计服务CTS接入LTS**
APIG模板	根据APIG资料中提供的日志字段被定义。	否	否	**API网关APIG接入LTS**
DCS审计日志	根据DCS资料中提供的日志字段被定义。	否	否	**分布式缓存服务DCS接入LTS**

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
TOMCAT	根据TOMCAT官网提供的字段名称进行nginx解析的名称。	否	否	TOMCAT结构化模板字段介绍
NGINX	根据NGINX资料中提供的日志字段被定义。	否	否	NGINX结构化模板字段介绍
GAUSSV5审计日志	根据GAUSSV5资料中提供的日志字段被定义。	否	否	云数据库GaussDB接入LTS
DDS审计日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
DDS错误日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
DDS慢日志	根据DDS资料中提供的日志字段被定义。	否	否	文档数据库服务DDS接入LTS
CFW访问控制日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
CFW攻击日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
CFW流量日志	根据CFW资料中提供的日志字段被定义。	否	否	云防火墙CFW接入LTS
MYSQL错误日志	根据MYSQL资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS
MYSQL慢日志	根据MYSQL资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
POSTGRESQL慢日志	根据POSTGRESQL慢日志资料中提供的日志字段被定义。	否	否	云数据库RDS for PostgreSQL接入LTS
POSTGRESQL错误日志	根据POSTGRESQL错误日志资料中提供的日志字段被定义。	否	否	云数据库RDS for PostgreSQL接入LTS
SQLSERVER错误日志	根据SQLSERVER资料中提供的日志字段被定义。	否	否	云数据库RDS for SQLServer接入LTS
GeminiDB Redis慢日志	根据GeminiDB Redis资料中提供的日志字段被定义。	否	否	云数据库GeminiDB接入LTS
CDN	根据CDN资料中提供的日志字段被定义。	否	否	内容分发网络CDN接入LTS
SMN	根据SMN资料中提供的日志字段被定义。	否	否	消息通知服务SMN接入LTS
GAUSSDB_MYSQL错误日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	云数据库GaussDB(for MySQL)接入LTS
GAUSSDB_MYSQL慢日志	根据GAUSSDB_MYSQL资料中提供的日志字段被定义。	否	否	云数据库GaussDB(for MySQL)接入LTS
ER企业路由器	根据ER企业路由器资料中提供的日志字段被定义。	否	否	企业路由器ER接入LTS
MYSQL审计日志	根据MYSQL审计日志资料中提供的日志字段被定义。	否	否	云数据库RDS for MySQL接入LTS

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
GeminiDB Cassandra慢日志	根据GeminiDB Cassandra慢日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Cassandra接入LTS
GeminiDB Mongo慢日志	根据GeminiDB Mongo慢日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Mongo接入LTS
GeminiDB Mongo错误日志	根据GeminiDB Mongo错误日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB Mongo接入LTS
WAF访问日志	根据WAF访问日志资料中提供的日志字段被定义。	否	否	Web应用防火墙WAF接入LTS
WAF攻击日志	根据WAF攻击日志资料中提供的日志字段被定义。	否	否	Web应用防火墙WAF接入LTS
DMS重平衡日志	根据DMS重平衡日志资料中提供的日志字段被定义。	否	否	分布式消息服务Kafka版接入LTS
CCE审计日志	根据CCE审计日志资料中提供的日志字段被定义。	否	否	云容器引擎CCE应用日志接入LTS
CCE事件日志	根据CCE事件日志资料中提供的日志字段被定义。	否	否	云容器引擎CCE应用日志接入LTS
GeminiDB Redis审计日志	根据GeminiDB Redis审计日志资料中提供的日志字段被定义。	否	否	云数据库GeminiDB接入LTS
influx慢日志	根据influx慢日志资料中提供的日志字段被定义。	否	否	-
METRIC系统日志	根据日志生成指标任务过滤的监控日志字段被定义。	否	否	-

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除	字段详情介绍
Microgateway	根据Microgateway应用网关提供的日志字段被定义。	否	否	-
GeminiDB Mongo接口审计日志	根据GeminiDB Mongo接口审计日志提供的日志字段被定义。	否	否	-

----结束

自定义模板

在“选择模板”下，选择“自定义模板”，选择已有的结构化模板。模板来源有以下两种方式：

- 在配置正则分析、JSON、分隔符或Nginx方式时单击左下角的“另存为模板”，系统会弹出“另存模板”页面，输入模板名称，单击“确定”，完成自定义模板的保存，会在“自定义模板”下的模板列表看到该模板。
- 新增结构化模板，具体操作如下：

  在“选择模板”下，选择“自定义模板”，单击“新增结构化模板”，在“新增结构化模板”界面选择正则分析、JSON、分隔符或Nginx方式，进行配置，配置完成后输入模板名称，单击“确定”。完成自定义模板的保存，会在“自定义模板”下的模板列表看到该模板。

6.3.4 设置云端结构化字段

使用限制

结构化字段最大长度为16KB，超过部分会被截断。

设置结构化字段

在进行结构化配置字段提取之后，可对结构化字段进行设置，具体设置规则如下表。

表 6-8 结构化字段设置规则

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除
正则分析（自动生成）	用户自定义。 名称必须以字母开始，且仅包含字母和数字。	是	是
正则分析（手动输入）	• 支持在输入正则表达式时进行命名。 • 支持使用系统默认命名field1、field2、field3……，或对其修改后的名称。	是	是

日志提取方式	字段名称	字段类型是否可修改	字段是否可删除
JSON格式	智能提取字段名称，可定义别名。	是	是
分隔符	默认名称field1、field2、field3……，可进行修改。	是	是
Nginx	根据Nginx配置生成，可定义别名。	是	是
自定义模板	用户自定义。	是	是

说明

正则分析（手动输入）、JSON格式、分隔符、Nginx和自定义模板的字段名称需要满足如下要求：

- 只支持输入英文、数字、中划线、下划线及小数点。
- 不能以小数点、下划线开头或以小数点结尾。
- 长度为1-64个字符。

设置 tag 字段

设置结构化配置时，可以对日志维度信息进行tag字段设置，设置完成后可以在可视化界面对设置字段进行SQL查询。

步骤1 在字段提取步骤中选择“tag字段”页签。

步骤2 单击“添加字段”。

步骤3 在tag字段列表中“字段名称”，输入需要设置 tag字段名称，例如hostIP。

说明

tag字段功能上线前设置的结构化配置，在修改结构化配置进行tag字段设置时，系统tag不会带出示例字段。

步骤4 如需添加多个字段可单击“添加字段”，继续添加。

步骤5 设置完成后单击“保存”。

说明

- tag支持的系统字段包括：category、clusterId、clusterName、containerName、hostIP、hostId、hostName、nameSpace、pathFile、podName。
- tag不支持的系统字段包括：groupName、logStream、lineNum、content、logContent、logContentSize、collectTime。
- 日志提取字段和tag字段可以同时设置。

----结束

6.3.5 设置云端结构化自定义日志时间

当日志接入云日志服务（LTS）时，您可以通过开启“自定义日志时间”开关，将日志中的时间字段设置为接入配置的时间。

开启自定义日志时间

步骤1 在左侧导航栏中选择“日志管理”。

步骤2 结构化日志以日志流为单位，在“日志管理”页面选择目标日志组和日志流。

步骤3 单击日志流名称进入日志流详情页面，单击右上角  ，在弹出页面中，选择“云端结构化解析”，进入日志结构化配置页面，选择对应的日志提取方法进行配置。

步骤4 配置完成后，开启自定义日志时间开关  ，配置如下参数。

表 6-9 参数配置表

参数	说明	示例
字段key	已提取字段的名称。单击下拉框选择已提取的字段，该字段为string或long类型。	test
字段value	已提取的字段value，选择字段key后，将自动填充。	2022-07-19 12:12:00
时间格式	请参考**常见日志时间格式**。	yyyy-MM-dd HH:mm:ss
操作	单击“校验”，提示“时间格式和字段value匹配成功”则表示校验成功。	-

说明

切换自定义日志时间开关时，可能会导致日志搜索界面在切换时间点附近出现时间偏差，请勿频繁切换自定义日志时间开关。

----结束

常见日志时间格式

支持的常见日志时间格式如下表所示。

说明

默认情况下，日志服务中的日志时间戳精确到秒，所以时间格式只需配置到秒，无需配置毫秒、微秒等信息。

表 6-10 时间格式

时间格式	说明	示例
EEE	星期的缩写。	Fri
EEEE	星期的全称。	Friday
MMM	月份的缩写。	Jan
MMMM	月份的全称。	January

时间格式	说明	示例
dd	每月第几天，十进制，范围为01~31。	07, 31
HH	小时，24小时制。	22
hh	小时，12小时制。	11
MM	月份，十进制，范围为01~12。	08
mm	分钟，十进制，范围为00~59。	59
a	AM或PM。	AM、PM
hh:mm:ss a	12小时制的时间组合。	11:59:59 AM
HH:mm	小时和分钟组合。	23:59
ss	秒数，十进制，范围为00~59。	59
yy	年份，十进制，不带世纪，范围为00~99。	04、98
yyyy	年份，十进制。	2004、1998
d	每月第几天，十进制，范围为1~31。 如果是个位数字，前面需要加空格。	7、31
DDD	一年中的天数，十进制，范围为001~366。	365
u	星期几，十进制，范围为1~7，1表示周一。	2
w	每年的第几周，星期天是一周的开始，范围为00~53。	23
w	每年的第几周，星期一是一周的开始，范围为01~53。 如果一月份刚开始的一周>=4天，则认为是第1周，否则认为下一个星期是第1周。	24
U	星期几，十进制，范围为0~6，0代表周日。	5
EEE MMM dd HH:mm:ss yyyy	标准的日期和时间。	Tue Nov 20 14:12:58 2020
EEE MMM dd yyyy	标准的日期，不带时间。	Tue Nov 20 2020
HH:mm:ss	标准的时间，不带日期。	11:59:59
%s	Unix时间戳。	147618725

示例

常见的时间标准、示例及对应的时间表达式如下所示。

表 6-11 示例

示例	时间表达式	时间标准
2022-07-14T19:57:36+08:00	yyyy-MM-dd'T'HH:mm:ssXXX	自定义
1548752136	%s	自定义
27/Jan/2022:15:56:44	dd/MMM/yyyy:HH:mm:ss	自定义
2022-08-15 17:53:23+08	yyyy-MM-dd HH:mm:ssX	自定义
2022-08-05T08:24:15.536+0000	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-20T10:04:03.204000Z	yyyy-MM-dd'T'HH:mm:ss.SSSZ	自定义
2022-08-22T06:52:08Z	yyyy-MM-dd'T'HH:mm:ssZ	自定义
2022-07-24T10:06:41.000	yyyy-MM-dd'T'HH:mm:ss.SSS	自定义
[2022-12-11 15:05:07.012]	[yyyy-MM-dd HH:mm:ss.SSS]	自定义
Monday, 02-Jan-06 15:04:05 MST	EEEE, dd-MMM-yy HH:mm:ss Z	RFC850
Mon, 02 Jan 2006 15:04:05 MST	EEE, dd MMM-yyyy HH:mm:ss Z	RFC1123
02 Jan 06 15:04 MST	dd MMM yy HH:mm Z	RFC822
02 Jan 06 15:04 -0700	dd MMM yy HH:mm Z	RFC822Z
2023-01-02T15:04:05.999999999Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339Nano
2023-01-02T15:04:05Z07:00	yyyy-MM-dd'T'HH:mm:ss Z	RFC3339
2022-12-11 15:05:07	yyyy-MM-dd HH:mm:ss	自定义

6.4 搜索语法与功能

6.4.1 搜索语法

云日志服务LTS提供一套搜索语法用于设置搜索条件，帮助您更有效地搜索日志。

查询语句用来指定日志查询时的过滤规则，返回符合条件的日志。根据索引配置方式可分为全文查询和字段查询，根据查询精确程度可分为精确查询和模糊查询。详细请参考表6-12。

SQL是用于访问和处理数据库的标准计算机语言。管道符提供了使用SQL语句进行数据分析的能力，具体请参考SQL92语法标准。

说明

使用SQL语句进行查询时，建议SQL语句中的字段名或别名使用双引号，例如：select "filed1.a" as "别名" from log。

表 6-12 查询方式

查询方式	说明	示例
全文查询	配置全文索引后，日志服务根据您设置的分词符将整条日志拆分成多个词。您可以指定关键字（字段名、字段值）和查询规则进行查询。	hello and world表示查询同时包含关键字hello和world的日志。
字段查询	配置字段索引后，您可以指定字段名称和字段值（Key:Value）进行查询。根据字段索引中设置的数据类型，您可以进行多种类型的基础查询和组合查询。	time>60 and region:r*表示查询time字段值大于60且region字段值以r开头的日志。
精确查询	使用完整的词进行查询。日志服务查询采用的是分词法，精确查询时并不能完全匹配关键词。例如查询语句为abc def，查询结果将包含所有abc和def的日志，无法完全匹配目标短语。如果您要完全匹配短语abc def，可以使用短语查询。更多信息，请参见短语查询、如何精准查询日志。	region:r1表示查询region字段值精确匹配r1的日志。
模糊查询	在查询语句中指定一个64个字符以内的词，在词的中间或者末尾加上模糊查询关键字，即星号（*）或问号（?），日志服务会在所有日志中为您查询到符合条件的100个词，返回包含这100个词并满足查询条件的所有日志。指定的词越精确，查询结果越精确。	my*表示在所有日志中查找以my开头的100个词，并返回包含这些词的日志。

运算符

查询语句支持如下运算符。

表 6-13 运算符

运算符	说明
and	and运算符。例如request_method:GET and status:200。
AND	与运算符，等同于and。
or	or运算符。例如request_method:GET or status:200。
OR	或运算符，等同于or。
not	not运算符。例如request_method:GET not status:200、not status:200。
( )	用于提高括号内查询条件的优先级。例如(request_method:GET or request_method:POST) and status:200。
:	用于字段查询（Key:Value），例如request_method:GET。 **说明** 如果字段名称（key）或者字段值（value）内有空格或冒号（:）等保留字符，请使用双引号（""）包裹字段名称（key）或者字段值（value）。例如： • **"request method":GET** • **message:"This is a log"** • **time:"09:00:00"** • **ipv6:"2024:AC8:2ac::d09"**
""	使用双引号（""）包裹一个语法关键词，可以将该语法关键词转换成普通字符。例如"and"表示查询包含and的日志，此处的and不代表运算符。在字段查询中双引号（""）内的所有词被当成一个整体。
\	转义符号，用于转义双引号（""），转义后的引号表示符号本身。例如日志内容为instance_id:nginx"01"，您可以使用instance_id:nginx\"01\"进行查询。 **说明** 转义双引号\"在搜索语句中必须成对出现。例如支持使用fieldName : "\"error\""或fieldName:"\\"，不支持使用fieldName : "error\""或者fieldName : "error\"
*	通配符查询，匹配零个、单个、多个字符。例如host:aliyund*c。
?	通配符查询，匹配单个字符。例如host:aliyund?c。
>	查询某字段值大于某数值的日志。例如request_time>100。
>=	查询某字段值大于或等于某数值的日志。例如request_time>=100。
<	查询某字段值小于某数值的日志。例如request_time<100。
<=	查询某字段值小于或等于某数值的日志。例如request_time<=100。

运算符	说明
=	查询某字段值等于某数值的日志。针对double、long类型的字段，等号（=）和冒号（:）作用相同。例如request_time=100等同于request_time:100。
in	查询某字段值处于某数值范围内的日志，中括号表示闭区间，小括号表示开区间，两个数字之间使用空格分隔。例如request_time in [100 200]或request_time in (100 200]。
#""	用于搜索包含目标短语的日志，可以保证关键词出现的顺序。 **说明** 短语搜索中的星号（*）和问号（?）会被视为普通字符，因此短语搜索不支持模糊搜索，可以用来搜索日志中的星号（*）和问号（?）。

6.4.2 短语搜索

短语搜索用于准确匹配目标短语，例如搜索语句**abc def**，不区分先后顺序，将匹配所有同时包含**abc**和**def**的日志。短语搜索和关键词搜索的区别请参考**表6-14**。

- 短语搜索：在关键词搜索语法的基础上实现，短语搜索能够区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。短语搜索适用于英文短语、中文短语的搜索，不支持模糊搜索。
- 关键词搜索：关键词搜索是基于分词实现，通过分词符先将搜索内容拆分为多个关键词，然后匹配日志。关键词搜索不会区分多个关键词在日志中出现的顺序，因此只要日志中按照搜索的与或非逻辑能命中关键词，该日志就会被搜索到。

表 6-14 搜索区别

搜索方式	短语搜索	关键词搜索
搜索区别	区分关键词的顺序，用于精准匹配目标短语，搜索结果更加精确。	不区分关键词的顺序，按照搜索逻辑命中关键词即可。
举例说明	假设您的日志流中存在2条原始日志，如下： • 原始日志1：this service is lts • 原始日志2：lts is service	
	短语搜索：#"is lts"，会命中1条日志。	关键词搜索：is lts，会命中2条日志。
	短语搜索：#"lts is"，会命中1条日志。	关键词搜索：lts is，会命中2条日志。

搜索语法

表 6-15 搜索方式

搜索方式	说明
全文搜索	• #"abc def" • content:#"abc def" **说明** content为日志原文对应的内置字段，**#"abc def"**等同于**content:#"abc def"**，默认匹配日志原文的内容。
字段搜索	key:#"abc def" **说明** • value参数不可为空。 • 字段搜索和not运算符配合使用时，还会匹配到不包含该字段的日志。

使用限制

- 短语搜索不支持搭配模糊搜索。

  短语搜索中的星号（*）和问号（？）会被视为普通字符，因此短语搜索不支持搭配模糊搜索，可以用来搜索日志中的星号（*）和问号（？）。
- 短语搜索不支持对分词符进行搜索。

  例如搜索语句**#"var/log"**，其中 / 为分词符，搜索语句等同于**#"var log"**，会搜索包含目标短语**var log**的日志。同理，搜索语句**#"var:log"**、**#"var;log"**等搜索的也是包含目标短语**var log**的日志。
- 中文搜索推荐采用短语搜索。

  由于中文默认采用的是一元分词，每个汉字单独分词，搜索时会匹配同时包含搜索语句中每一个汉字的日志，本身便具有模糊搜索的特性，当需要更加精确的结果时，推荐采用短语搜索。

示例说明

表 6-16 搜索说明

搜索需求	搜索语句
搜索User-Agent字段值包含短语Mon, 17 Apr 2023的日志。	User-Agent:#"Mon, 17 Apr 2023"
搜索User-Agent字段值包含短语Mozilla/5.0的日志。	User-Agent:#"Mozilla/5.0"
搜索week字段值包含短语星期一的日志。	week:#"星期一"

6.4.3 查看 LTS 实时日志

日志接入LTS后，日志每隔大约1分钟上报一次，在实时日志页面，您最多需要等待1分钟左右，即可查看实时上报的日志，实现对日志数据的快速检索与分析。

- 若实时日志大于1MB且总数小于2000条时时，LTS会清空前500KB的日志，保留最新的500KB日志。
- 若实时日志未超过1MB，但总数超过2000条时，LTS会清空前1000条日志，保留最新的1000条日志。

说明

正常情况下，每隔5秒加载一次。如果这5秒内没有产生日志，则不显示；5秒后会继续调用接口，刷新出产生的日志数据。即如果每5秒都有日志数据产生，则加载数据时延为5秒。

前提条件

- 已创建日志组和日志流。
- 已完成ICAgent安装。
- 已配置日志采集规则。

查看 LTS 实时日志

如果您正在使用实时查看功能，请停留在实时查看页面，请勿切换页面。如果离开实时查看页面，实时查看功能将会停止，重新开启后上一次查看的实时日志将不会显示。

步骤1 在云日志服务管理控制台，单击“日志管理”。

步骤2 在日志组列表中，单击日志组名称前对应的 。

步骤3 在日志流列表中，单击日志流名称，进入日志详情页面。

步骤4 在“实时日志”页签，查看实时日志。

说明

通过来源类型分别筛选主机和K8S的日志。

- 来源类型选择主机时， 设置主机IP和文件路径。
- 来源类型选择K8S时，设置实例名称、容器名称和文件路径。
- 字段过滤：从索引配置、结构化配置、最新日志获取。

日志每隔大约1分钟上报一次，在日志消息区域，您最多需要等待1分钟左右，即可查看实时上报的日志。

同时，还可以通过页面右上方的“清屏”、“暂停”对日志消息区域进行操作。

- 清屏：清除日志消息区域已经显示出来的日志。
- 暂停：暂停日志消息的实时显示，页面定格在当前已显示的日志。

  暂停后，“暂停”会变成“继续”，再次单击“继续”，日志消息继续实时显示。

----结束

6.4.4 快速分析

日志包含了系统性能及业务等信息，例如关键词ERROR的多少反应了系统的健康度，关键词BUY的多少反应了业务的成交量等，当您需要了解这些信息时，可以通过快速分析功能，指定查询日志关键词，LTS能够针对您配置的关键词进行统计，并生成指标数据，以便您实时了解系统性能及业务等信息。

说明

- 支持对前100000条日志进行分析。

  快速分析的目的是快速返回字段值的分布情况和变化趋势，并没有对全量数据进行分析，是一种采样结果。
- 支持通过查询时间和查询条件过滤日志进行分析。

  快速分析是对通过查询语句查询到的日志进行分析，当查询到的日志数目为零时，快速分析没有结果。
- 支持将快速分析生成查询语句。

  单击快速分析的某一行分析结果，可自动生成查询语句，查询日志并生成新的快速分析。
- 快速分析的字段长度最大为2000字节。
- 快速分析字段分布统计展示前100条数据。

前提条件

快速分析的对象为结构化日志中提取的关键字段，创建快速分析前请先对原始日志进行**结构化配置**。

创建快速分析

可通过日志结构化打开“快速分析”按钮进行创建。也可通过如下步骤进行创建。

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 快速分析以日志流为单位，请在“日志管理”页面选择目标日志组和日志流。

步骤3 支持两种方式创建快速分析：

1. 单击 进入设置详情页面，在索引配置页签的字段索引下方，添加字段时开启快速分析。
2. 在云端结构化解析页签，开启自动配置索引和快速分析，默认是开启状态。开启后将使用结构化字段配置字段索引并打开快速分析。

步骤4 在搜索分析页签，单击“创建快速分析”，跳转到索引配置页面添加需要快速分析的字段。

步骤5 单击“确定”，快速分析创建完成。

图 6-3 查看快速分析

说明

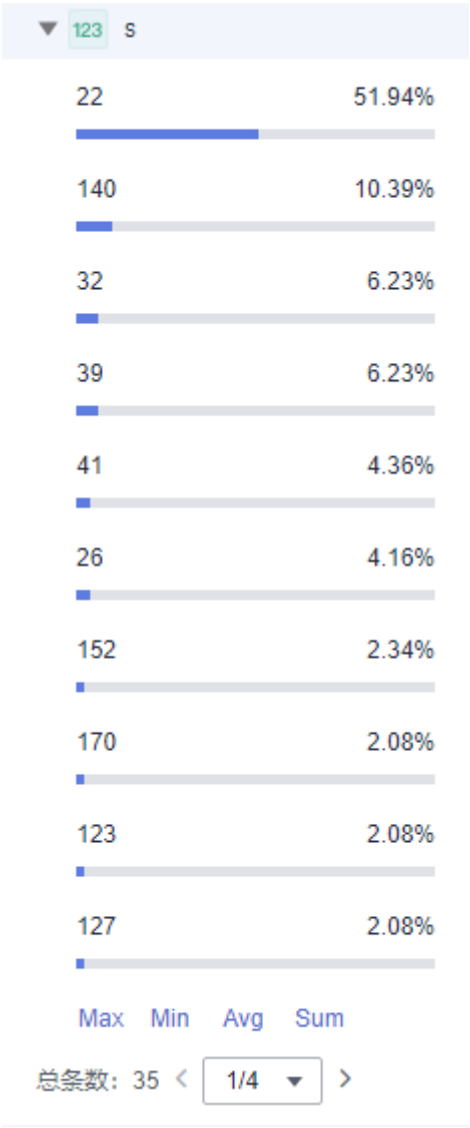
- abc 表示String类型字段。
- 1.2 表示float类型字段。
- 123 表示long类型字段。
- 快速分析的字段长度最大为2000字节。
- 快速分析字段展示前100条数据。
- 单击 即可查看一键生成的图表展示，string类型的字段支持展示字段分布值统计和智能聚合时间折线图，long和float数值类型的字段只支持展示智能聚合时间折线图。单击图表即可进入详情页面。
- 单击字段分布值统计或智能聚合时间折线图，会自动跳转到可视化界面并生成对应的SQL查询语句进行查询，更加直观地展示字段值的分布和变化趋势。更多信息请参见**可视化**。

----结束

二次分析

快速分析功能支持对float类型和long类型的字段进行二次分析，具体如下：

快速统计最大项、最小项、平均值和总和，分别单击目标字段下的**Max**、**Min**、**Avg**、**Sum**，快速查找所有项中的最大项、最小项、平均值和总和。



6.4.5 快速查询

当您需要重复使用某一关键字搜索日志时，可以将其设置为快速查询语句。

创建快速查询

1. 在云日志服务控制台，单击“日志管理”。
2. 在日志组列表中，单击日志组名称前对应的 按钮。
3. 在日志流列表中，单击日志流名称，进入日志详情页面。
4. 在搜索分析页签，单击 ，输入“快速查询名称”和“快速查询语句”。默认开启快速查询和快速查询（保存本地）。

图 6-4 创建快速查询

- 快速查询名称，用于区分多个快速查询语句。名称自定义，需要满足如下要求：
  - 只支持输入英文、数字、中文、中划线、下划线及小数点。
  - 不能以小数点、下划线开头或以小数点结尾。
  - 长度为1-64个字符。
- 快速查询语句，搜索日志时需要重复使用的关键字，例如“error*”。

5. 单击“确定”，完成快速查询条件的创建。在左侧导航栏的快速查询页签，即可查看到保存成功的语句。

   单击快速查询语句的名称，查看日志详情。

查看上下文

您可以通过本操作查看指定日志生成时间点前后的日志，用于在运维过程中快速定位问题。

1. 在搜索分析页签下方的原始日志页面，单击可以查看上下文。

   在查看上下文结果中，可以查看该日志的前后若干条日志详细信息。
2. 在弹出的查看上下文页面中，查看日志上下文。

表 6-17 查看上下文日志功能介绍

功能	说明
查询行数	根据需要选择查询日志的行数。
高亮显示	输入需要高亮的字符串，回车确认，在日志内容中高亮显示。

功能	说明
过滤日志	输入需要过滤的字符串，回车确认，在日志内容中高亮显示。当高亮显示和过滤日志同时设置时，均可高亮显示。
显示字段	查看上下文，默认字段为content，单击“显示字段”选择查看其他字段的上下文。
更早	从当前位置往前查看设置**查询行数**的二分之一。例如：当查询行数设置为100时，单击“更早”则从当前位置朝前显示50行，此时行号为-50；再次单击“更早”，依次叠加分别为-100、-150、-200.....
当前位置	当前日志位置。当设置了更早或更新时，单击“当前位置”可回到查看上下文开始的位置，即行数为0时。
更新	从当前位置往后查看设置**查询行数**的二分之一。例如：当查询行数设置为100时，单击“更新”则从当前位置朝后显示50行，此时行号为50；再次单击“更新”，依次叠加分别为100、150、200.....
简洁模式	打开简洁模式，只显示行号和content内容。关闭简洁模式，显示日志详情。
下载	目前仅支持下载content字段内容到本地查看。

6.5 SQL 函数

目前此功能处于邀测中，暂不支持用户开通，邀测期间可能会偶现查询慢，持续优化中。

6.5.1 聚合函数

本文介绍聚合函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-18 聚合函数

函数	描述
approx_distinct函数	用于估算x中不重复值的个数。
approx_percentile函数	用于对x进行正序排列，返回处于percentage位置的数值。
arbitrary函数	用于返回x中任意一个非空的值。
max_by函数	查询y为最大值时对应的x值，或查询最大的n个y值对应的x值。
min_by函数	查询y为最小值时对应的x值，或查询最小的n个y值对应的x值。

函数	描述
count函数	用于计数。
max函数	用于查询x中最大的值。
min函数	用于查询x中最小值。
avg函数	用于计算x的算术平均值。

approx_distinct 函数

用于估算x中不重复值的个数。

- 估算x中不重复值的个数，默认存在2.3%的标准误差
  语法：approx_distinct(x)
- 估算x中不重复值的个数，支持自定义标准误差
  语法：approx_distinct(x, e)

表 6-19 参数说明

参数名称	描述	类型	是否必选
x	原始字段	任意	是
e	自定义标准误差	double类型，取值为[0.0115, 0.26]	否

返回值类型：bigint类型

- 示例1：使用approx_distinct函数估算不重复的clientIp字段值，标准误差为2.3%。

  表 6-20 查询分析结果

类型	场景
查询语句	SELECT approx_distinct(clientIp)
返回结果	1

- 示例2：使用approx_distinct函数估算不重复的clientIp字段值， 自定义标准误差为10%。

表 6-21 查询分析结果

类型	场景
查询语句	SELECT approx_distinct(clientIp, 0.1)
返回结果	1

approx_percentile 函数

用于对x进行正序排列，返回处于percentage位置的数值。

- 对于x进行正序排列，返回处于percentage位置的x，返回结果为double类型。

  语法：approx_percentile(x, percentage)
- 对x进行正序排列，返回处于percentage01、percentage02位置的x，返回结果为array(double,double)类型。

  语法：approx_percentile(x, array[percentage01, percentage02)]
- 对x和权重的乘积进行正序排列，返回大约处于percentage位置的x，返回结果为double类型。

  语法：approx_percentile(x, weight, percentage)
- 对x和权重的乘积进行正序排列，返回处于percentage01、percentage02位置的x，返回结果为array(double,double)类型。

  语法：approx_percentile(x, weight, array[percentage01, percentage02...])
- 对x和权重的乘积进行正序排列，返回大约处于percentage位置的x，返回结果为double类型。支持设置返回结果的准确度。

  语法：approx_percentile(x, weight, percentage, accuracy)

表 6-22 参数说明

参数名称	描述	类型	是否必选
x	原始字段	double	是
percentage	百分比值，取值范围为[0,1]。	double	是
weight	权重，大于1的整数。 设置权重后，系统根据x与权重的乘积进行排序。	int	否
accuracy	准确度，取值范围为(0,1)。	double	否

返回值类型：double类型或array(double,...,double)类型

- 示例1：对request_time列进行排序后，返回大约处于50%位置的request_time字段的值。

表 6-23 查询分析结果

类型	场景
查询语句	SELECT approx_percentile(request_time, 0.5)
返回结果	45.0

- 示例2：对request_time列进行排序后，返回处于10%、20%及70%位置的request_time字段的值。

表 6-24 查询分析结果

类型	场景
查询语句	SELECT approx_percentile(request_time,array[0.1,0.2,0.7])
返回结果	[17.0, 24.0, 59.0]

- 示例3：根据request_time与权重的乘积对request_time列进行排序后，返回大约处于50%位置的request_time字段的值, 权重值为60。

表 6-25 查询分析结果

类型	场景
查询语句	SELECT approx_percentile(request_time, 60, 0.5)
返回结果	45.0

- 示例4：根据request_time与权重的乘积对request_time列进行排序后，返回大约处于80%和90%位置的request_time字段的值，权重值为60。

表 6-26 查询分析结果

类型	场景
查询语句	SELECT approx_percentile(request_time, 60, array[0.8, 0.9])
返回结果	[66.0,73.0]

- 示例5：根据request_time与权重的乘积对request_time列进行排序后，返回大约处于50%位置的request_time字段的值，权重值为60，准确度为0.2。

表 6-27 查询分析结果

类型	场景
查询语句	SELECT approx_percentile(request_time, 60, 0.5, 0.2)

类型	场景
返回结果	45.0

arbitrary 函数

用于返回x中任意一个非空的值。

语法：arbitrary(x)

表 6-28 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意	是

返回值类型：与参数值的数据类型一致。

示例：select arbitrary(region)

表 6-29 查询分析结果

类型	场景
查询语句	arbitrary(region)
返回结果	r2

max_by 函数

查询y为最大值时对应的x值，或查询最大的n个y值对应的x值。

- 查询y为最大值时对应的x值。
  语法：max_by(x, y)
- 查询最大的n个y值对应的x值。
  语法：max_by(x, y, n)

表 6-30 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意数据类型	是
y	原始字段。	任意数据类型	是
n	大于0的整数。	int	否

返回值类型：与参数值的数据类型一致。

- 示例1：统计请求时长最大时对应的请求方法。

  表 6-31 查询分析结果

类型	场景
查询语句	SELECT max_by(request_method, request_time)
返回结果	GET

- 示例2：统计请求时长最大的3个请求对应的请求方法。

  表 6-32 查询分析结果

类型	场景
查询语句	SELECT max_by(request_method, request_time, 3)
返回结果	["GET","GET","GET"]

min_by 函数

查询y为最小值时对应的x值，或查询最小的n个y值对应的x值。

- 查询y为最小值时对应的x值。
  语法：max_by(x, y)
- 查询最小的n个y值对应的x值。
  语法：max_by(x, y, n)

表 6-33 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意数据类型	是
y	原始字段。	任意数据类型	是
n	大于0的整数。	int	否

返回值类型：与参数值的数据类型一致

- 示例1：统计请求时长最小时对应的请求方法。

表 6-34 查询分析结果

类型	场景
查询语句	SELECT min_by(request_method, request_time)
返回结果	POST

- 示例2：统计请求时长最小的3个请求对应的请求方法。

表 6-35 查询分析结果

类型	场景
查询语句	SELECT min_by(request_method, request_time, 3)
返回结果	["POST","POST","POST"]

count 函数

用于计数。

- 统计所有的日志条数。
  语法：COUNT(*)
- 统计所有的日志条数。等同于count(*)。
  语法：COUNT(1)
- 统计x中值不为NULL的日志条数。
  语法：COUNT(x)

表 6-36 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意	是

返回值类型：int

示例：select COUNT(*)

表 6-37 查询分析结果

类型	场景
查询语句	COUNT(*)
返回结果	1

max 函数

用于查询x中最大的值。

- 查询x中最大的值。
  语法：max(x)
- 查询x中最大的n个值，结果返回为数组。
  语法： max(x, n)

表 6-38 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意数据	是
n	返回最大值的个数	正整数	否

返回值类型：与参数值的数据类型一致。

- 示例1：查询x中的最大值

  表 6-39 查询分析结果

类型	场景
查询语句	SELECT max(x)
返回结果	99.0

- 示例2：查询x中的最大的2个值

  表 6-40 查询分析结果

类型	场景
查询语句	SELECT max(x, 2)
返回结果	[99.0, 99.0]

min 函数

用于查询x中最小值。

- 查询x中最小的值。
  语法：min(x)
- 查询x中最小的n个值，结果返回为数组。
  语法： min(x, n)

表 6-41 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	任意数据	是
n	返回最小值的个数	正整数	否

返回值类型：与参数值的数据类型一致。

- 示例1：查询x中的最小值

  表 6-42 查询分析结果

类型	场景
查询语句	SELECT min(x)
返回结果	10.0

- 示例2：查询x中的最小的2个值

  表 6-43 查询分析结果

类型	场景
查询语句	SELECT min(x, 2)
返回结果	[10.0, 10.0]

avg 函数

用于计算x的算术平均值。

语法：AVG(x)

表 6-44 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	number	是

返回值类型：double

示例：select AVG(value)

表 6-45 查询分析结果

类型	场景
查询语句	AVG(value)
返回结果	1

6.5.2 字符串函数

本文介绍字符串函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-46 字符串函数

函数	描述
lpad函数	在字符串的开头填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。
rpad函数	在字符串的结尾填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。
replace函数	将字符串中的匹配字符串替换成指定的字符串，如果未指定替换字符串，则将匹配的字符串从原来的字符串中删除。
reverse函数	将字符串转换成反向顺序的字符串。
repeat函数	将字符串重复指定次数。
contains_string函数	判断字符串是否包含指定字符串。
icontains_string函数	判断字符串是否不包含指定字符串。
textcat函数	将两个字符串拼接起来，返回拼接后的结果。
btrim函数	删除字符串左右两侧的空格。
parse_long函数	将字符串转换成基于指定基数的Long型，如果未指定基数，则基于十进制对原始字符串进行转换。
split函数	使用指定的分隔符拆分字符串，并返回子字符串的集合。如果设置了limit，则使用limit限制拆分字符串的数量。
split_part函数	使用指定的分隔符拆分字符串，并返回指定位置的字符串。如果拆分后指定的位置超过数组的长度，则返回“ ”。
split_to_map函数	使用指定的第一个分隔符拆分字符串，然后使用指定的第二个分隔符第二次拆分字符串，返回第二次拆分后的结果。
string_format函数	返回以Java的String.format方式格式化的字符串。

函数	描述
strpos函数	查询目标子串初次出现在字符串中的位置，如果目标子串未出现在字符串中，则返回0。
substr函数	使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串，同substring。
substring函数	使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串。
length函数	如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。
char_length函数	与length函数功能一致，如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。
character_length函数	与length函数功能一致。
strlen函数	与length函数功能一致。
levenshtein_distance函数	计算两个字符串str1和str2之间的最小编辑距离。
normalize函数	格式化字符串。
to_utf8函数	将字符串转换为UTF-8编码格式。
chr函数	将ascii值转换成字符。
concat函数	用于将多个参数拼接成一个字符串。

lpad 函数

在字符串的开头填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。

语法：lpad(str, len, lpadStr)

表 6-47 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
len	结果字符串的长度。 • 如果原始字符串的长度小于len，那么在字符串的开头填充指定的字符。 • 如果原始字符串的长度大于len，那么只返回字符串的len个字符。	int	是

参数名称	描述	类型	是否必选
lpadStr	填充字符。	String	否

返回值类型：String类型

示例：SELECT LPAD('hello world', 10), LPAD('hello', 10, 'e')

表 6-48 查询分析结果

类型	场景1	场景2
查询语句	LPAD('hello world', 10)	LPAD('hello', 10, 'e')
返回结果	hello worl	eeeeehello

rpad 函数

在字符串的结尾填充指定字符，直到指定长度后返回结果字符串。如果未指定填充字符，则使用“ ”进行填充。

语法：rpad(str, len, rpadStr)

表 6-49 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
len	结果字符串的长度。 • 如果原始字符串的长度小于len，那么在字符串的结尾填充指定的字符。 • 如果原始字符串的长度大于len，那么只返回字符串的len个字符。	Integer	是
rpadStr	填充字符。	String	否

返回值类型：String类型

示例：SELECT RPAD('hello world', 10), RPAD('hello', 10, 'e')

表 6-50 查询分析结果

类型	场景1	场景2
查询语句	RPAD('hello world', 10)	RPAD('hello', 10, 'e')

类型	场景1	场景2
返回结果	hello worl	helloeeeee

replace 函数

将字符串中的匹配字符串替换成指定的字符串，如果未指定替换字符串，则将匹配的字符串从原来的字符串中删除。

语法：replace(str, subStr, replaceStr)

表 6-51 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
subStr	目标子串。	String	是
replaceStr	用于替换的字符串。	String	否

返回值类型：String类型

示例：SELECT REPLACE('hello world', 'o'), REPLACE('hello world', 'w', 'new w')

表 6-52 查询分析结果

类型	场景1	场景2
查询语句	REPLACE('hello world', 'o')	REPLACE('hello world', 'w', 'new w')
返回结果	hell wrld	hello new world

reverse 函数

将字符串转换成反向顺序的字符串。

语法：reverse(str)

表 6-53 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

返回值类型：String类型

示例：SELECT REVERSE('hello world')

表 6-54 查询分析结果

类型	场景
查询语句	REVERSE('hello world')
返回结果	dlrow olleh

repeat 函数

将字符串重复指定次数。

语法：repeat(str, num)

表 6-55 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
num	重复次数	Integer	是

返回值类型：String类型

示例：SELECT REPEAT('hello world', 2), REPEAT('hello world', 0)

表 6-56 查询分析结果

类型	场景1	场景2
查询语句	REPEAT('hello world', 2)	REPEAT('hello world', 0)
返回结果	hello worldhello world	

contains_string 函数

判断字符串是否包含指定字符串。

语法：contains_string(str, containsStr)

表 6-57 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
containsStr	指定字符串。	String	是

返回值类型：Boolean类型

示例：SELECT CONTAINS_STRING('hello world', 'llo')

表 6-58 查询分析结果

类型	场景
查询语句	CONTAINS_STRING('hello world', 'llo')
返回结果	true

icontains_string 函数

判断字符串是否不包含指定字符串。

语法：icontains_string(str, containsStr)

表 6-59 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
containsStr	指定字符串。	String	是

返回值类型：Boolean类型

示例：SELECT ICONTAINS_STRING('hello world', 'llo')

表 6-60 查询分析结果

类型	场景
查询语句	ICONTAINS_STRING('hello world', 'llo')
返回结果	false

textcat 函数

将两个字符串拼接起来，返回拼接后的结果。

语法：textcat(str1, str2)

表 6-61 参数说明

参数名称	描述	类型	是否必选
str1	待拼接字符串1。	String	是
str2	待拼接字符串2。	String	是

返回值类型：String类型

示例：SELECT TEXTCAT('hello ', 'world')

表 6-62 查询分析结果

类型	场景
查询语句	TEXTCAT('hello ', 'world')
返回结果	hello world

btrim 函数

删除字符串左右两侧的空格。

语法：btrim(str)

表 6-63 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

返回值类型：String类型

示例：SELECT BTRIM(' hello world ')

表 6-64 查询分析结果

类型	场景
查询语句	BTRIM(' hello world ')
返回结果	hello world

parse_long 函数

将字符串转换成基于指定基数的Long型，如果未指定基数，则基于十进制对原始字符串进行转换。

语法：parse_long(str, radix)

表 6-65 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
radix	基数。	Integer	否

返回值类型：Long类型

示例：SELECT PARSE_LONG('-1234', 8), PARSE_LONG('1234')

表 6-66 查询分析结果

类型	场景1	场景2
查询语句	PARSE_LONG('-1234', 8)	PARSE_LONG('1234')
返回结果	-668	1234

split 函数

使用指定的分隔符拆分字符串，并返回子字符串的集合。如果设置了limit，则使用limit限制拆分字符串的数量。

语法：split(str, splitStr, limit)

表 6-67 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
splitStr	分隔符。	String	是
limit	基数。	Integer	否

返回值类型：ARRAYS<STRING>类型

示例：SELECT SPLIT('helloworld', 'o'), SPLIT('helloworld', 'o', 2)

表 6-68 查询分析结果

类型	场景1	场景2
查询语句	SPLIT('helloworld','o')	SPLIT('helloworld', 'o', 2)
返回结果	["hell","w","rld"]	["hell","world"]

split_part 函数

使用指定的分隔符拆分字符串，并返回指定位置的字符串。如果拆分后指定的位置超过数组的长度，则返回“ ”。

语法：split_part(str, splitStr, position)

表 6-69 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
splitStr	分隔符。	String	是
position	需要返回的字符串的位置。	Integer	是

返回值类型：STRING类型

示例：SELECT SPLIT_PART('helloworld', 'o', 2), SPLIT_PART('helloworld', 'o', 4)

表 6-70 查询分析结果

类型	场景1	场景2
查询语句	SPLIT_PART('helloworld','o', 2)	SPLIT_PART('helloworld', 'o', 4)
返回结果	w	

split_to_map 函数

使用指定的第一个分隔符拆分字符串，然后使用指定的第二个分隔符第二次拆分字符串。返回第二次拆分后的结果。

语法：split_to_map(str, splitStr1, splitStr2)

表 6-71 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
splitStr1	分隔符1。	String	是
splitStr2	分隔符2。	String	是

返回值类型：ARRAY类型

示例：SELECT SPLIT_TO_MAP('upstream_response_time:123, request_time:456', ',', ':')

表 6-72 查询分析结果

类型	场景
查询语句	SPLIT_TO_MAP('upstream_response_time:123, request_time:456', ',', ':')
返回结果	{"request_time":"456","upstream_response_time":"123"}

string_format 函数

返回以Java的String.format方式格式化的字符串。

语法：string_format(str, Object... args)

表 6-73 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
args	参数。	String/Integer/Long/Boolean/Double	是

返回值类型：String类型

示例：SELECT STRING_FORMAT('My name is %s and I am %d years old.', 'name', age)

表 6-74 查询分析结果

类型	场景
查询语句	STRING_FORMAT('My name is %s and I am %d years old.', 'Tom', 25)
返回结果	My name is Tom and I am 25 years old.

说明

%d：Output integer.
%f：Output floating point number.
%s：Output character string.
%n：Output newline character.
%c：Output Characters
%b：Output bool type
%e：Output Exponential Type
%tc：Output Date and time information
%tF：Output format is YYY-MM-DD.
%tr：Output in HH:MM:SS PM format
可以参考Java的String.format获取更多的细节

strpos 函数

查询目标子串初次出现在字符串中的位置，如果目标子串未出现在字符串中，则返回0。

语法：strpos(str, subStr)

表 6-75 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
subStr	填充字符。	String	是

返回值类型：Integer类型

示例：SELECT STRPOS('hello world', 'llo'), STRPOS('llo', 'hello world')

表 6-76 查询分析结果

类型	场景1	场景2
查询语句	STRPOS('hello world', 'llo')	STRPOS('llo', 'hello world')
返回结果	3	0

substr 函数

使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串，同substring。

语法：substr(str, start, length)

表 6-77 参数说明

参数名称	描述	类型	是否必选
str	原始字符串	String	是
start	开始提取子串的位置，从1开始	Integer	是
length	子串的长度	Integer	否

返回值类型：String类型

示例：SELECT SUBSTR('helloworld', 5), SUBSTR('helloworld', 5, 3)

表 6-78 查询分析结果

类型	场景1	场景2
查询语句	SUBSTR ('helloworld', 5)	SUBSTR ('helloworld', 5, 3)
返回结果	oworld	owo

substring 函数

使用起始位置和长度返回子字符串。如果没有长度参数，则返回从起始位置开始的整个字符串。

语法：substring(str, start, length)

表 6-79 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是
start	开始提取子串的位置，从1开始。	Integer	是
length	子串的长度。	Integer	否

返回值类型：String类型

示例：SELECT SUBSTRING('helloworld', 5), SUBSTRING('helloworld', 5, 3)

表 6-80 查询分析结果

类型	场景1	场景2
查询语句	SUBSTRING('helloworld', 5)	SUBSTRING('helloworld', 5, 3)

类型	场景1	场景2
返回结果	oworld	owo

length 函数

如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。

语法：length(str)

表 6-81 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

返回值类型：Integer类型

示例：SELECT LENGTH('HELLO WORLD')

表 6-82 查询分析结果

类型	场景1
查询语句	LENGTH('HELLO WORLD')
返回结果	11

char_length 函数

与length函数功能一致，如果字符串表达式是字符数据类型，则返回输入的字符长度；否则，返回字符串表达式的字节长度（不小于位数除以8的最小整数）。

语法：char_length(str)

表 6-83 参数说明

参数名称	描述	类型	是否必选
str	原始字符串	String	是

返回值类型：Integer类型

示例：SELECT CHAR_LENGTH('HELLO WORLD')

表 6-84 查询分析结果

类型	场景
查询语句	CHAR_LENGTH('HELLO WORLD')
返回结果	11

character_length 函数

与length函数功能一致。

语法：character_length(str)

表 6-85 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

返回值类型：Integer类型

示例：SELECT CHARACTER_LENGTH('HELLO WORLD')

表 6-86 查询分析结果

类型	场景
查询语句	CHARACTER_LENGTH('HELLO WORLD')
返回结果	11

strlen 函数

与length函数功能一致。

语法：strlen(str)

表 6-87 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

返回值类型：Integer类型

示例：SELECT STRLEN('HELLO WORLD')

表 6-88 查询分析结果

类型	场景
查询语句	STRLEN('HELLO WORLD')
返回结果	11

levenshtein_distance 函数

计算两个字符串str1和str2之间的最小编辑距离。

语法：levenshtein_distance(str1, str2)

表 6-89 参数说明

参数名称	描述	类型	是否必选
str1	原始字符串1。	String	是
str2	原始字符串2。	String	是

返回值类型：Integer类型

示例：SELECT LEVENSHTEIN_DISTANCE('horse', 'ros')

表 6-90 查询分析结果

类型	场景
查询语句	LEVENSHTEIN_DISTANCE('horse', 'ros')
返回结果	3

normalize 函数

格式化字符串str。

- 使用NFC格式格式化字符串str。
  语法：normalize(str)
- 使用给定格式格式化字符串str，支持的格式有NFC, NFD, NFKC, NFKD.
  语法：normalize(str, form)

表 6-91 参数说明

参数名称	描述	类型	是否必选
str	原始字符串。	String	是

参数名称	描述	类型	是否必选
form	格式化类型。	String	否

返回值类型：String类型

示例：SELECT NORMALIZE('schön'), NORMALIZE('Henry \u2163', 'nfd')

表 6-92 查询分析结果

类型	场景1	场景2
查询语句	NORMALIZE('schön')	NORMALIZE('Henry \u2163', 'nfd')
返回结果	schön	Henry Ⅳ

to_utf8 函数

将字符串转换为UTF-8编码格式。

语法：to_utf8(str)

表 6-93 参数说明

参数名称	描述	类型	是否必选
str1	原始字符串1	String	是
str2	原始字符串2	String	是

返回值类型：Integer类型

示例：SELECT TO_UTF8('中')

表 6-94 查询分析结果

类型	场景
查询语句	TO_UTF8('中')
返回结果	5Lit

chr 函数

将ascii值转换成字符。

语法：chr(ascii)

表 6-95 参数说明

参数名称	描述	类型	是否必选
ascii	ascii码	Integer	是

返回值类型：String类型

示例：SELECT CHR(99)

表 6-96 查询分析结果

类型	场景
查询语句	CHR(99)
返回结果	c

concat 函数

用于将多个参数拼接成一个字符串。

语法：concat(x, y...)

表 6-97 参数说明

参数名称	描述	类型	是否必选
x,y...	原始字段。	string	是

返回值类型：String类型

示例：select concat('1','+','1')

表 6-98 查询分析结果

类型	场景
查询语句	concat('1','+','1')
返回结果	1+1

6.5.3 日期和时间函数

本文介绍日期、时间函数的语法规则，包括参数解释、函数示例等。管道符特性处理的日期和时间格式为yyyy-MM-dd HH:mm:ss.SSS。

函数列表

表 6-99 日期和时间函数

函数	描述
current_timestamp函数	返回当前日期和时间，格式为yyyy-MM-dd HH:mm:ss.SSS
from_iso8601_date函数	将ISO8601格式的日期表达式expr转换为date类型的日期表达式，格式为yyyy-MM-dd
from_iso8601_timestamp函数	将ISO8601格式的日期表达式expr转换为timestamp类型的日期表达式，格式为yyyy-MM-dd HH:mm:ss.SSS
mills_to_timestamp函数	将UNIX时间戳转换为日期和时间表达式。
from_unixtime函数	将UNIX时间戳转换为日期和时间表达式。与mills_to_timestamp函数用法一致。
to_iso8601函数	将日期类型或时间戳类型的日期时间表达式转换为ISO8601格式的日期时间表达式。
timestamp_to_mills函数	将时间戳类型的日期和时间表达式转换为UNIX时间戳。
to_unixtime函数	将时间戳类型的日期和时间表达式转换为UNIX时间戳。与timestamp_to_mills函数用法一致。
time_ceil函数	将时间戳舍入，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。指定Origin作为时间戳，以设置舍入的参考时间。例如，TIME_CEIL(time, 'PT1H', '2016-06-27 00:30:00')测量的小时周期从00:30-01:30而不是00:00-01:00。时区（如果提供）应该是时区名称，如“America/Los_Angeles”或偏移量，如“-08:00”。此功能与ceil_to类似，但更灵活。
time_floor函数	向下舍入时间戳，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。指定Origin作为时间戳，以设置舍入的参考时间。例如，TIME_FLOOR(time, 'PT1H', '2016-06-27 00:30:00')测量的小时周期从00:30-01:30而不是00:00-01:00。时区（如果提供）应该是时区名称，如“America/Los_Angeles”或偏移量，如“-08:00”。此功能与floor_to类似，但更灵活。
ceil函数	使用时间单位对时间戳进行四舍五入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、MONTH、 QUARTER或 YEAR。
floor函数	使用时间单位对时间戳进行向下舍入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、MONTH、 QUARTER或 YEAR。

函数	描述
time_shift函数	将时间戳expr移动一个Period（步长时间）。Period可以是任何ISO8601的Period。
timezone_hour函数	计算系统时区和utc时区的小时偏移量。如果提供了时区，则计算系统时区与给定时区的偏移量。
timezone_minute函数	计算系统时区和utc时区的分钟偏移量。如果提供了时区，则计算系统时区与给定时区的偏移量。
time_format函数	将时间戳类型的日期和时间表达式转换为指定格式的日期和时间表达式。
date_format函数	将时间戳类型的日期和时间表达式转换为指定格式的日期和时间表达式。
time_parse函数	将日期和时间字符串转换为指定格式的时间戳类型的日期和时间表达式。
date_parse函数	将日期和时间字符串转换为指定格式的时间戳类型的日期和时间表达式。
time_extract函数	通过指定字段提取日期时间表达式的日期或时间部分。EPOCH, SECOND, MINUTE, HOUR, DAY(月的日), DOW（周的日）, DOY（年的日）, WEEK（年周）, MONTH（1到12）, QUARTER（1到4）,或YEAR,时区（如果提供）应为时区名称，如"America/Los_Angeles"或偏移量，如"-08:00"
date_trunc函数	根据您指定的时间单位截断日期和时间表达式，并以毫秒、秒、分钟、小时、天、月或年为单位对齐。
date_diff函数	返回时间戳expr1和时间戳expr2之间的单位数（有符号）。
current_date函数	返回当前日期，格式是“yyyy-MM-dd”。
now函数	返回当前日期和时间，格式是“yyyy-MM-dd HH:mm:ss.SSS”。功能与current_timestamp一致
date_add函数	给时间加上给定的时间间隔
current_time函数	返回当前时间，格式为HH:mm:ss.SSSSSS。
current_timezone函数	返回当前时区。
localtime函数	返回本地时间。
localtimestamp函数	返回本地的日期和时间。
year_of_week函数	返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。
yow函数	返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。

函数	描述
time_series函数	用于补全您查询时间窗口内缺失的数据。

current_timestamp 函数

返回当前日期和时间，格式为yyyy-MM-dd HH:mm:ss.SSS。

语法：current_timestamp()

返回值类型：String类型

示例：SELECT CURRENT_TIMESTAMP()

表 6-100 查询分析结果

类型	场景
查询语句	CURRENT_TIMESTAMP()
返回结果	2023-04-17 10:48:41.286

from_iso8601_date 函数

将ISO8601格式的日期表达式转化为date类型的日期表达式，格式为YYYY-MM-DD。

语法：from_iso8601_date(expr)

表 6-101 参数说明

参数名称	描述	类型	是否必选
expr	ISO8601格式的日期表达式	String	是

返回值类型：String类型

示例：SELECT FROM_ISO8601_DATE('2018-05-14')

表 6-102 查询分析结果

类型	场景
查询语句	FROM_ISO8601_DATE('2018-05-14')
返回结果	2018-05-14

from_iso8601_timestamp 函数

将ISO8601格式的日期和时间表达式转化为timestamp类型的日期和时间表达式，格式为YYYY-MM-DD HH:MM:SS.Ms Time_zone。

语法：from_iso8601_timestamp(expr)

表 6-103 参数说明

参数名称	描述	类型	是否必选
expr	ISO8601格式的日期和时间表达式	String	是

返回值类型：timestamp类型。

示例：SELECT FROM_ISO8601_TIMESTAMP('2018-05-14T11:51:50.153+08:00')

表 6-104 查询分析结果

类型	场景
查询语句	FROM_ISO8601_TIMESTAMP('2018-05-14T11:51:50.153+08:00')
返回结果	2018-05-14 11:51:50.153 +08:00

mills_to_timestamp 函数

将UNIX时间戳转换为日期和时间表达式。

- 将UNIX时间戳转化为没有时区的时间戳类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS。

  语法：mills_to_timestamp(expr)

  表 6-105 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是

  返回值类型：String类型

  示例：SELECT MILLS_TO_TIMESTAMP(1626774758000)

表 6-106 查询分析结果

类型	场景
查询语句	MILLS_TO_TIMESTAMP(1626774758000)
返回结果	2021-07-20 09:52:38.000

- 将UNIX时间戳转化为带时区的timestamp类型的日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TimeZone。

  语法：mills_to_timestamp(expr, timezone)

  表 6-107 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是
timezone	时区	String	是

  返回值类型：String类型

  示例：SELECT MILLS_TO_TIMESTAMP(1626774758000, 'Asia/Shanghai')

  表 6-108 查询分析结果

类型	场景
查询语句	MILLS_TO_TIMESTAMP(1626774758000, 'Asia/Shanghai')
返回结果	2021-07-20 17:52:38.000 Asia/Shanghai

- 将UNIX时间戳expr转换为具有时区的时间戳类型的日期和时间表达式。表达式中，hour和minute表示时区偏移量，格式为yyyy-MM-dd HH:mm:ss.SSS 时区偏移量。

  语法：mills_to_timestamp(expr, hour, minutes)

  表 6-109 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是
hour	小时	Integer	是
minute	分钟	Integer	是

  返回值类型：String类型

示例：SELECT MILLS_TO_TIMESTAMP(1626774758000, -2, 0)

表 6-110 查询分析结果

类型	场景
查询语句	MILLS_TO_TIMESTAMP(1626774758000, -2, 0)
返回结果	2021-07-20 07:52:38.000-02:00

from_unixtime 函数

将UNIX时间戳转换为日期和时间表达式。与mills_to_timestamp函数用法一致。

- 将UNIX时间戳转化为没有时区的时间戳类型的日期和时间表达式。
  语法：from_unixtime(expr)

  表 6-111 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是

  返回值类型：String
  示例：SELECT FROM_UNIXTIME(1626774758000)

  表 6-112 查询分析结果

类型	场景
查询语句	FROM_UNIXTIME(1626774758000)
返回结果	2021-07-20 09:52:38.000

- 将UNIX时间戳转化为带时区的timestamp类型的日期和时间表达式。
  语法：from_unixtime(expr, timezone)

  表 6-113 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是
timezone	时区	String	是

  返回值类型：String类型
  示例：SELECT FROM_UNIXTIME(1626774758000, 'Asia/Shanghai')

表 6-114 查询分析结果

类型	场景
查询语句	FROM_UNIXTIME(1626774758000, 'Asia/Shanghai')
返回结果	2021-07-20 17:52:38.000 Asia/Shanghai

- 将UNIX时间戳expr转换为具有时区的时间戳类型的日期和时间表达式。表达式中，hour和minute表示时区偏移量。

  语法：from_unixtime(expr, hour, minutes)

  表 6-115 参数说明

参数名称	描述	类型	是否必选
expr	UNIX时间戳	Long	是
hour	小时	Integer	是
minute	分钟	Integer	是

  返回值类型：String

  示例：SELECT FROM_UNIXTIME(1626774758000, -2, 0)

  表 6-116 查询分析结果

类型	场景
查询语句	FROM_UNIXTIME(1626774758000, -2, 0)
返回结果	2021-07-20 07:52:38.000 -02:00

to_iso8601 函数

将日期类型或时间戳类型的日期时间表达式转换为ISO8601格式的日期时间表达式。

语法：to_iso8601(expr)

表 6-117 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd或yyyy-MM-dd HH:mm:ss.SSS	String类型	是

返回值类型：String类型

示例：SELECT TO_ISO8601('2023-04-15'), TO_ISO8601('2023-04-15 11:13:35.954')

表 6-118 查询分析结果

类型	场景1	场景2
查询语句	TO_ISO8601('2023-04-15')	TO_ISO8601('2023-04-15 11:13:35.954')
返回结果	2023-04-15	2023-04-15T11:13:35.954Z

timestamp_to_mills 函数

将时间戳类型的日期和时间表达式转换为UNIX时间戳。

语法：timestamp_to_mills(expr)

表 6-119 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String类型	是

返回值类型：Long类型

示例：SELECT TIMESTAMP_TO_MILLS('2023-04-12 16:15:22.285')

表 6-120 查询分析结果

类型	场景
查询语句	TIMESTAMP_TO_MILLS('2023-04-12 16:15:22.285')
返回结果	1681287322285

to_unixtime 函数

将时间戳类型的日期和时间表达式转换为UNIX时间戳。与timestamp_to_mills函数用法一致。

语法：to_unixtime(expr)

表 6-121 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是

返回值类型：Long类型

示例：SELECT TO_UNIXTIME('2023-04-12 16:15:22.285')

表 6-122 查询分析结果

类型	场景
查询语句	TO_UNIXTIME('2023-04-12 16:15:22.285')
返回结果	1681287322285

time_ceil 函数

将时间戳舍入，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。指定Origin作为时间戳，以设置舍入的参考时间。例如，TIME_CEIL(time, 'PT1H', '2016-06-27 00:30:00')测量的小时周期从00:30-01:30而不是00:00-01:00。时区（如果提供）应该是时区名称，如“America/Los_Angeles”或偏移量，如“-08:00”。此功能与ceil_to类似，但更灵活。

语法：time_ceil(expr, period, [origin, [timezone]])

说明

origi和timezone加中括号表示可选，非必填。

表 6-123 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
period	ISO8601的周期	String	是
origin	原始时间，格式为yyyy-MM-dd HH:mm:ss.SSS	String	否
timezone	时区	String	否

返回值类型：String类型

示例1：SELECT TIME_CEIL('2023-04-23 06:46:40.000', 'PT2H')

表 6-124 查询分析结果

类型	场景
查询语句	TIME_CEIL('2023-04-23 06:46:40.000', 'PT2H')
返回结果	2023-04-23 08:00:00.000

示例2：SELECT TIME_CEIL('2023-04-25 06:53:45.000', 'PT2H', '2012-12-18 01:20:21.000')

表 6-125 查询分析结果

类型	场景
查询语句	TIME_CEIL('2023-04-25 06:53:45.000', 'PT2H', '2012-12-18 01:20:21.000')
返回结果	2023-04-25 07:20:21.000

示例3：SELECT TIME_CEIL('2023-04-25 09:44:35.000', 'PT2H', '2012-12-18 04:11:11.000', 'Europe/Rome')

表 6-126 查询分析结果

类型	场景
查询语句	TIME_CEIL('2023-04-25 09:44:35.000', 'PT2H', '2012-12-18 04:11:11.000', 'Europe/Rome')
返回结果	2023-04-25 10:11:11.000

说明

Period描述：

period的格式应为P[n]Y[n]M[n]DT[n]H[n]M[n]S或P[n]W。在这些表示中，【n】是日期和时间元素的数量。不需要前导零，但每个元素的最大位数应由通信各方商定。大写字母P、Y、M、W、D、T、H、M和S是每个日期和时间元素的指示符，不被替换。

P是放置在持续时间表示的开始处的持续时间指示符（对于期间）。

Y是日历年数值之后的年指示符。

M是日历月数值之后的月份指示符。

W是周数值之后的周指示符。

D是日历天数值之后的日期指示符。

T是表示的时间分量之前的时间指示符。

H是小时数值之后的小时指示符。

M是分钟指示符，紧随分钟数的值。

S是秒数值之后的第二个指示符。

例如，“P3Y6M4DT12H30M5S”表示“三年六个月四天十二小时三十分五秒”的持续时间。

更多信息，请访问https://en.wikipedia.org/wiki/ISO_8601

time_floor 函数

向下舍入时间戳，将其作为新的时间戳返回。Period可以是任何ISO8601的周期，如P3M（季度）或PT12H（半天）。指定Origin作为时间戳，以设置舍入的参考时间。例如，TIME_FLOOR(time, 'PT1H', '2016-06-27 00:30:00')测量的小时周期从00:30-01:30而不是00:00-01:00。时区（如果提供）应该是时区名称，如“America/Los_Angeles”或偏移量，如“-08:00”。此功能与floor_to类似，但更灵活。

语法：time_floor(expr, period, [origin, [timezone]])

说明

origi和timezone加中括号表示可选，非必填。

表 6-127 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
period	ISO8601的周期	String	是
origin	原始时间，格式为yyyy-MM-dd HH:mm:ss.SSS	String	否
timezone	时区	String	否

返回值类型：String类型

示例1：SELECT TIME_FLOOR('2023-04-23 06:46:40.000', 'PT2H')

表 6-128 查询分析结果

类型	场景
查询语句	TIME_FLOOR('2023-04-23 06:46:40.000', 'PT2H')
返回结果	2023-04-23 06:00:00.000

示例2：SELECT TIME_FLOOR('2023-04-25 06:53:45.000', 'PT2H', '2012-12-18 01:20:21.000')

表 6-129 查询分析结果

类型	场景
查询语句	TIME_FLOOR('2023-04-25 06:53:45.000', 'PT2H', '2012-12-18 01:20:21.000')
返回结果	2023-04-25 05:20:21.000

示例3：SELECT TIME_FLOOR('2023-04-23 06:46:40.000', 'PT2H', '2012-12-18 04:11:11.000', 'Europe/Rome')

表 6-130 查询分析结果

类型	场景
查询语句	TIME_FLOOR('2023-04-25 09:44:35.000', 'PT2H', '2012-12-18 04:11:11.000', 'Europe/Rome')
返回结果	2023-04-25 08:11:11.000

ceil 函数

使用时间单位对时间戳进行四舍五入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、 MONTH、 QUARTER或 YEAR。

语法：ceil(expr, unit)

表 6-131 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
unit	时间单位	String	是

返回值类型：String类型

示例：SELECT CEIL('2023-04-20 11:28:31.770', 'DAY')

表 6-132 查询分析结果

类型	场景
查询语句	CEIL ('2023-04-20 11:28:31.770', 'DAY')
返回结果	2023-04-21 00:00:00.000

floor 函数

使用时间单位对时间戳进行向下舍入，单位可以是SECOND、MINUTE、 HOUR、 DAY、 WEEK、 MONTH、 QUARTER或 YEAR。

语法：floor(expr, unit)

表 6-133 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是

参数名称	描述	类型	是否必选
unit	时间单位	String	是

返回值类型：Strin类型

示例：SELECT FLOOR('2023-04-20 11:28:31.770', 'DAY')

表 6-134 查询分析结果

类型	场景
查询语句	FLOOR('2023-04-20 11:28:31.770', 'DAY')
返回结果	2023-04-20 00:00:00.000

time_shift 函数

将时间戳expr移动一个Period（步长时间）。Period可以是任何ISO8601的Period。

语法：time_shift(expr, period, step, [timezone])

表 6-135 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
period	ISO8601的周期	String	是
step	步长	Integer	是
timezone	时区	String	否

返回值类型：String类型

示例：SELECT TIME_SHIFT('2023-04-22 15:31:30.000', 'P1D', 5), TIME_SHIFT('2023-04-22 15:31:30.000', 'P1D', 5, 'Asia/Shanghai')

表 6-136 查询分析结果

类型	场景1	场景2
查询语句	TIME_SHIFT('2023-04-22 15:31:30.000', 'P1D', 5)	TIME_SHIFT('2023-04-22 15:31:30.000', 'P1D', 5, 'Asia/Shanghai')
返回结果	2023-04-27 15:31:30.000	2023-04-27 23:31:30.000

timezone_hour 函数

计算时区的小时偏移量。

语法：timezone_hour(expr)

表 6-137 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TIMEZONE	String	否

返回值类型：Integer类型

示例：SELECT TIMEZONE_HOUR('2023-04-22 15:31:30.000 Europe/Rome')

表 6-138 查询分析结果

类型	场景
查询语句	TIMEZONE_HOUR('2023-04-22 15:31:30.000 Europe/Rome')
返回结果	1

timezone_minute 函数

计算时区的分钟偏移量。

语法：timezone_minute(expr)

表 6-139 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS TIMEZONE	String	否

返回值类型：Integer类型

示例：SELECT TIMEZONE_MINUTE('2023-04-22 15:31:30.000 Europe/Rome')

表 6-140 查询分析结果

类型	场景
查询语句	TIMEZONE_MINUTE('2023-04-22 15:31:30.000 Europe/Rome')

类型	场景
返回结果	0

time_format 函数

将时间戳类型的日期和时间表达式转换为指定格式的日期和时间表达式。

语法：time_format(expr, format)

表 6-141 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式。	String	是
format	时间格式。	String	是

返回值类型：String类型

示例：SELECT TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss')

表 6-142 查询分析结果

类型	场景
查询语句	TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss')
返回结果	2023-02-16 07:38:25

说明

format描述：

%a Abbreviation for the week. For example, Sun and Sat.

%b Abbreviation of the month. For example, Jan and Dec.

%c Month. Numeral type. Range: 1-12.

%D Day of the month. The value must be suffixed, for example, 0th, 1st, 2nd, and 3rd.

%d Day of the month. The value ranges from 01 to 31 in decimal notation.

%e Day of the month. The value ranges from 1 to 31 in decimal notation.

%H Hour, 24-hour system.

%h Hour, 12-hour system.

%i Minute. Numeral type. Range: 00–59.

%j Day of the year. The value ranges from 001 to 366.

%k Hour. The value ranges from 0 to 23.

%l Hour. The value ranges from 1 to 12.

%M The English expression of the month, for example, January, December.

%m Month. Numeral format. The value ranges from 01 to 12.

%p AM and PM.

%r Time in the 12-hour format. The format is hh:mm:ss AM/PM.

%S Indicates the second. The value ranges from 00 to 59.

%s Indicates the second. The value ranges from 00 to 59.

%T Time, in the 24-hour format of hh:mm:ss.

%v The first week of the year, Monday is the first day of the week. The value ranges from 01 to 53.

%W The name of the day of the week. For example, Sunday and Saturday.

%w Day of the week. Sunday is day 0.

%Y A 4-digit year, for example, 2020.

%y A 2-digit year, for example, 20.

%% Escape character for%.

date_format 函数

将时间戳类型的日期和时间表达式转换为指定格式的日期和时间表达式。与time_format函数功能一致。

语法：date_format(expr, format)

表 6-143 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
format	时间格式	String	是

返回值类型：String类型

示例：SELECT DATE_FORMAT(current_timestamp(),'%H%i')

表 6-144 查询分析结果

类型	场景
查询语句	DATE_FORMAT(current_timestamp(),'%H%i')
返回结果	1432

说明

format描述：

%a Abbreviation for the week. For example, Sun and Sat.

%b Abbreviation of the month. For example, Jan and Dec.

%c Month. Numeral type. Range: 1-12.

%D Day of the month. The value must be suffixed, for example, 0th, 1st, 2nd, and 3rd.

%d Day of the month. The value ranges from 01 to 31 in decimal notation.

%e Day of the month. The value ranges from 1 to 31 in decimal notation.

%H Hour, 24-hour system.

%h Hour, 12-hour system.

%i Minute. Numeral type. Range: 00–59.

%j Day of the year. The value ranges from 001 to 366.

%k Hour. The value ranges from 0 to 23.

%l Hour. The value ranges from 1 to 12.

%M The English expression of the month, for example, January, December.

%m Month. Numeral format. The value ranges from 01 to 12.

%p AM and PM.

%r Time in the 12-hour format. The format is hh:mm:ss AM/PM.

%S Indicates the second. The value ranges from 00 to 59.

%s Indicates the second. The value ranges from 00 to 59.

%T Time, in the 24-hour format of hh:mm:ss.

%v The first week of the year, Monday is the first day of the week. The value ranges from 01 to 53.

%W The name of the day of the week. For example, Sunday and Saturday.

%w Day of the week. Sunday is day 0.

%Y A 4-digit year, for example, 2020.

%y A 2-digit year, for example, 20.

%% Escape character for%.

time_parse 函数

time_parse将给定的字符串expr，依据用户自定义的pattern参数以 **Joda DateTimeFormat**模式解析为时间戳。若<pattern>未填写则以默认的ISO8601将字符串解析。

语法：time_parse(expr, pattern)

1. 字段样例

   __time: 2023-02-16T07:38:25.306Z

start_time:2023-02-14 02:35:56

2. 查询和分析语句
```
SELECT __time,TIME_PARSE(start_time,'yyyy-MM-dd HH:mm:ss')
```
3. 查询和分析结果

表 6-145 查询和分析结果

__time	_col0	_col1
2023-02-16T07:38:25.306Z	2023-02-14T02:35:56.000Z	2023-02-16 07:38:25

date_parse 函数

将日期和时间字符串转换为指定格式的时间戳类型的日期和时间表达式。

语法：date_parse(expr, pattern)

表 6-146 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式。	String	是
pattern	日期和时间表达式的转换格式	String	是

返回值类型：timestamp类型

示例：SELECT DATE_PARSE('1996-03-22 14:32:00.000','%Y-%m-%d %h:%i')

表 6-147 查询分析结果

类型	场景
查询语句	DATE_PARSE('1996-03-22 14:32:00.000','%Y-%m-%d %h:%i')
返回结果	1996-03-22 14:32:00.000

说明

pattern在设置时，如果设置了分钟，那么就必须同时配置分钟之前的年、月、日和时。

time_extract 函数

通过指定字段提取日期时间表达式的日期或时间部分。EPOCH, SECOND, MINUTE, HOUR, DAY(月的日), DOW（周的日）, DOY（年的日）, WEEK（年周）, MONTH（1到12）, QUARTER（1到4）,或YEAR,时区（如果提供）应为时区名称，如"America/Los_Angeles"或偏移量，如"-08:00"

语法：time_extract(expr, unit)

表 6-148 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
unit	时间单位	String	是

返回值类型：Long类型

示例：SELECT TIME_EXTRACT('2023-05-05','YEAR')

表 6-149 查询分析结果

类型	场景
查询语句	TIME_EXTRACT('2023-05-05','YEAR')
返回结果	2023

date_trunc 函数

根据您指定的时间单位截断日期和时间表达式，并以毫秒、秒、分钟、小时、天、月或年为单位对齐。

语法：date_trunc(unit, expr)

表 6-150 参数说明

参数名称	描述	类型	是否必选
expr	日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
unit	时间单位	String	是

返回值类型：Long类型

示例：SELECT DATE_TRUNC('year', '2022-11-20 12:20:30.123')

表 6-151 查询分析结果

类型	场景
查询语句	DATE_TRUNC('year', '2022-11-20 12:20:30.123')
返回结果	2022-01-01 00:00:00.000

date_diff 函数

返回时间戳expr1和时间戳expr2之间的单位数（有符号）。

语法：date_diff(unit, expr1, expr2)

表 6-152 参数说明

参数名称	描述	类型	是否必选
expr1	第一个日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是
expr2	第二个日期和时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	
unit	时间单位	String	是

返回值类型：Long类型

示例：SELECT DATE_DIFF('MONTH', NOW(),'2022-11-20 12:20:30.123')

表 6-153 查询分析结果

类型	场景
查询语句	DATE_DIFF('MONTH', NOW(),'2022-11-20 12:20:30.123')
返回结果	4

current_date 函数

返回当前日期，格式是“yyyy-MM-dd”。

语法：current_date()

返回值类型：String类型

示例：SELECT CURRENT_DATE()

表 6-154 查询分析结果

类型	场景
查询语句	CURRENT_DATE()
返回结果	2023-04-17

now 函数

返回当前日期和时间，格式是“yyyy-MM-dd HH:mm:ss.SSS”。功能与current_timestamp一致

语法：now()

返回值类型：String类型

示例：SELECT NOW()

表 6-155 查询分析结果

类型	场景
查询语句	NOW()
返回结果	2023-04-17 10:48:41.286

date_add 函数

给时间加上给定的时间间隔。

语法：date_add(unit, n, expr)

表 6-156 参数说明

参数名称	描述	类型	是否必选
unit	时间单位，取值为millisecond、second、minute、hour、day、week、month、quarter、year	String	是
n	时间间隔	Long	是
expr	时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS	String	是

返回值类型：String类型

示例：SELECT DATE_ADD('minute', 7, '2023-05-15 16:14:00.569')

表 6-157 查询分析结果

类型	场景
查询语句	DATE_ADD('minute', 7, '2023-05-15 16:14:00.569')
返回结果	2023-05-15 16:21:00.569

current_time 函数

返回当前时间，格式为HH:mm:ss.SSSSSS。

语法：current_time()

返回值类型：String类型

示例：SELECT CURRENT_TIME()

表 6-158 查询分析结果

类型	场景
查询语句	CURRENT_TIME ()
返回结果	09:38:15.869881

current_timezone 函数

返回当前时区。

语法：current_timezone()

返回值类型：String类型

示例：SELECT CURRENT_TIMEZONE()

表 6-159 查询分析结果

类型	场景
查询语句	CURRENT_TIMEZONE ()
返回结果	Asia/Shanghai

localtime 函数

返回本地时间。

语法：localtime()

返回值类型：String类型

示例：SELECT LOCALTIME()

表 6-160 查询分析结果

类型	场景
查询语句	LOCALTIME()
返回结果	11:48:45.609726

localtimestamp 函数

返回本地的日期和时间。

语法：localtimestamp

返回值类型：String类型

示例：SELECT LOCALTIMESTAMP

表 6-161 查询分析结果

类型	场景
查询语句	LOCALTIMESTAMP
返回结果	2023-09-22 14:47:59.325

year_of_week 函数

返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。

语法：year_of_week(expr)

参数说明：

参数名称	描述	类型	是否必选
expr	时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS或yyyy-MM-dd	String	是

返回值类型：Integer类型

示例：SELECT YEAR_OF_WEEK('2023-09-22 14:47:59.325'), YEAR_OF_WEEK('2023-09-22')

表 6-162 查询分析结果

类型	场景1	场景2
查询语句	YEAR_OF_WEEK('2023-09-22 14:47:59.325')	YEAR_OF_WEEK('2023-09-22')
返回结果	2023	2023

yow 函数

返回目标日期在ISO周日历中的年份。year_of_week函数等同于yow函数。

语法：yow(expr)

表 6-163 参数说明

参数名称	描述	类型	是否必选
expr	时间表达式，格式为yyyy-MM-dd HH:mm:ss.SSS或yyyy-MM-dd	String	是

返回值类型：Integer类型

示例：SELECT YOW('2023-09-22 14:47:59.325'), YOW('2023-09-22')

表 6-164 查询分析结果

类型	场景1	场景2
查询语句	YOW('2023-09-22 14:47:59.325')	YOW('2023-09-22')
返回结果	2023	2023

time_series 函数

用于补全您查询时间窗口内缺失的数据。time_series函数必须搭配GROUP BY语法和ORDER BY语法使用。

语法：time_series(x, window_time, format, padding_data)

表 6-165 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	long或date	是
window_time	窗口大小，单位为s（秒）、m（分）、h（小时）、d（天）。例如2h、5m、3d。	String	是
format	返回结果的时间格式。	String	是
padding_data	补全的内容。包括：*0*：将缺失的值设置为0。*null*：将缺失的值设置为null。*last*：将缺失的值设置了上一个时间点对应的值。*next*：将缺失的值设置了下一个时间点对应的值。*avg*：将缺失的值设置为前后两个时间点的平均值。	String	是

返回值类型：String

示例：select time_series(__time__, '1d', 'HH:mm:ss','0') as time, count(*) as count group by time order by time asc

表 6-166 查询分析结果

类型	场景
查询语句	time_series(__time__, '1d', 'HH:mm:ss','0') as time, count(*) as count group by time order by time asc
返回结果	

6.5.4 正则式函数

本文介绍正则式函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-167 正则式函数

函数	描述
regexp_extract函数	提取目标字符串中符合正则表达式的子串。
regexp_like函数	判断目标字符串是否符合正则表达式。
regexp_extract_all函数	从目标字符串中提取符合正则表达式的子字符串。
regexp_replace函数	删除或替换目标字符串中符合正则表达式的子串。
regexp_split函数	分割目标字符串，返回被分割后的子串合集。

regexp_extract 函数

提取目标字符串中符合正则表达式的子串。

- 提取并返回目标字符串中符合正则表达式的第一个子串。
  语法：regexp_extract(expr, regularExpr)
- 提取并返回目标字符串中符合正则表达式的第n个子串。
  语法：regexp_extract(expr, regularExpr, n)

表 6-168 参数说明

参数名称	描述	类型	是否必选
expr	目标字符串。	String	是
regularExpr	包含捕获组的正则表达式，(\d)(\d)表示两个捕获组。	String	是
n	第n个符合正则表达式的子串。	Integer	否

返回值类型：String类型

示例：SELECT REGEXP_EXTRACT('HTTP/2.0', '\d+')

表 6-169 查询分析结果

类型	场景1	场景2
查询语句	REGEXP_EXTRACT('HTTP/2.0', '\d+')	REGEXP_EXTRACT ('HTTP/2.0', '\d+', 1)
返回结果	2	2

regexp_like 函数

判断目标字符串是否符合正则表达式。

语法：regexp_like(expr, regularExpr)

表 6-170 参数说明

参数名称	描述	类型	是否必选
expr	目标字符串	String	是
regularExpr	包含捕获组的正则表达式，(\d)(\d)表示两个捕获组	String	是

返回值类型：

Boolean类型

示例：SELECT REGEXP_LIKE('HTTP/2.0', '\d+')

表 6-171 查询分析结果

类型	场景
查询语句	REGEXP_LIKE('HTTP/2.0', '\d+')

类型	场景
返回结果	true

regexp_extract_all 函数

从目标字符串中提取符合正则表达式的子字符串。

语法：regexp_extract_all(expr, regularExpr)

表 6-172 参数说明

参数名称	描述	类型	是否必选
expr	目标字符串	String	是
regularExpr	包含捕获组的正则表达式，(\d)(\d)表示两个捕获组	String	是

返回值类型：Array类型

示例：SELECT REGEXP_EXTRACT_ALL('HTTP/2.0', '\d+')

表 6-173 查询分析结果

类型	场景
查询语句	REGEXP_EXTRACT_ALL ('HTTP/2.0', '\d+')
返回结果	["2","0"]

regexp_replace 函数

删除或替换目标字符串中符合正则表达式的子串。

- 删除目标字符串中符合正则表达式的子串，返回未被删除的子串。
  语法：regexp_replace(expr, regularExpr)
- 替换目标字符串中符合正则表达式的子串，返回被替换后的字符串。
  语法：regexp_replace(expr, regularExpr, replaceStr)

表 6-174 参数说明

参数名称	描述	类型	是否必选
expr	目标字符串	String	是
regularExpr	包含捕获组的正则表达式，(\d)(\d)表示两个捕获组	String	是

参数名称	描述	类型	是否必选
replaceStr	替换的字符串	String	否

返回值类型：String类型

示例：SELECT REGEXP_REPLACE('ab12cd34', '\d+'), REGEXP_REPLACE('ab12cd34', '\d+', '00')

表 6-175 查询分析结果

类型	场景1	场景2
查询语句	REGEXP_REPLACE('ab12cd34', '\d+')	REGEXP_REPLACE('ab12cd34', '\d+', '00')
返回结果	abcd	ab00cd00

regexp_split 函数

分割目标字符串，返回被分割后的子串合集。

语法：regexp_split(expr, regularExpr)

表 6-176 参数说明

参数名称	描述	类型	是否必选
expr	目标字符串	String	是
regularExpr	包含捕获组的正则表达式，(\d)(\d)表示两个捕获组	String	是

返回值类型：Array类型

示例：SELECT REGEXP_SPLIT('request_uri:/request/path-0/file-7','/')

表 6-177 查询分析结果

类型	场景
查询语句	REGEXP_SPLIT('request_uri:/request/path-0/file-7','/')
返回结果	["request_uri:","request","path-0","file-7"]

6.5.5 同比和环比函数

本文介绍同比和环比函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-178 同比和环比函数

函数	描述
compare函数	用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。
ts_compare函数	用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。

compare 函数

- 用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。
  语法：compare(x, n)
- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。
  语法：compare(x, n1, n2, n3...)

表 6-179 参数说明

参数名称	描述	类型	是否必选
x	待同比表达式。	double	是
n?	时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。	long	是

返回类型

JSON数组。格式为[当前计算结果,n秒前的计算结果,当前计算结果与n秒前计算结果的比值]。

示例说明

计算当前1小时和昨天同时段的访问量比值。

1. 选择查询和分析的时间范围为**1小时（整点时间）**，并执行如下查询和分析语句。其中**86400**表示当前时间减去86400秒(1天)。
   ```
   SELECT
   compare(PV, 86400)
   FROM  (SELECT count(*) AS PV )
   ```
2. 查询和分析结果。

   图 6-5 查询和分析结果

说明

- **5994.0**表示当前1小时（例如2021-01-02 00:00:00~2021-01-02 01:00:00）的网站访问量。
- **6000.0**表示昨天同时段（例如2021-01-01 00:00:00~2021-01-01 01:00:00）的网站访问量。
- **0.999**表示当前1小时与昨天同时段的网站访问量比值。

3. 分列显示查询和分析结果。

```
SELECT
diff[1] as "today",
diff[2] as "yesterday",
diff[3] as "ratio"
FROM(SELECT compare(pv, 86400) AS diff FROM (SELECT count(*) AS pv ))
```

图 6-6 查询和分析结果

today	yesterday	ratio
5993	6029	0.99402887

ts_compare 函数

ts_compare函数用于对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。

说明

ts_compare函数必须按照时间列进行分组（GROUP BY）。

语法格式

- 对比当前时间周期内的计算结果与n秒之前时间周期内的计算结果。

```
ts_compare(x, n)
```

- 对比当前时间周期内的计算结果与n1、n2、n3秒之前时间周期内的计算结果。

```
ts_compare(x, n1, n2, n3...)
```

表 6-180 环比函数参数说明

参数	说明
x	参数值为double类型或long类型。
n	时间窗口，单位为秒。例如3600（1小时）、86400（1天）、604800（1周）、31622400（1年）。

返回类型

JSON数组。格式为[当前计算结果, n秒前的计算结果, 当前计算结果与n秒前计算结果的比值, n秒前的UNIX时间戳]。

示例说明

环比今天3小时与昨天3小时的网站访问量。

选择查询和分析的时间范围为今天某3小时，并执行如下查询和分析语句。其中86400表示当前时间减去86400秒（1天），date_trunc('hour',__time)表示使用date_trunc函数将时间对齐到小时。

- 查询和分析语句

```
* | SELECT
  t_time,
  ts_compare(PV, 86400) AS data
FROM(
    SELECT
      to_unixtime(date_trunc('hour', __time)) AS t_time,
      count(*) AS PV
    GROUP BY
      t_time
    ORDER BY
      t_time
  )
GROUP BY
  t_time
```

- 查询和分析结果

t_time	data
2021-10-26T06:00:00.000Z	[159.0,224.0,0.7098214285714286,1.6351416E9]
2021-10-26T07:00:00.000Z	[100.0,148.0,0.6756756756756757,1.6351452E9]
2021-10-26T08:00:00.000Z	[100.0,100.0,1.0, 1.6016544E9, 1.6351488E9]

6.5.6 窗口函数

本文介绍窗口函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-181 窗口函数

函数	描述
ntile函数	用于将窗口分区内数据按照顺序分成N组。

ntile 函数

用于将窗口分区内数据按照顺序分成N组。

语法：

```
ntile(n) over (
   [partition by partition_expression]
   [order by order_expression]
 )
```

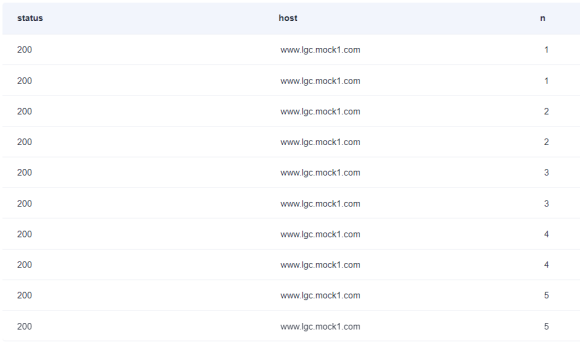
表 6-182 参数说明

参数名称	描述	类型	是否必选
n	组数。	int	是
partition by partition_expression	窗口分区，根据分区表达式将数据划分成不同的分区。	任意	是
order by order_expression	窗口排序，根据排序表达式对各个分区内的每一行进行排序。	任意	是

返回值类型：integer

示例：status,host,ntile(5) over (partition by status order by host) as n

表 6-183 查询分析结果

类型	场景
查询语句	status,host,ntile(5) over (partition by status order by host) as n
返回结果	

6.5.7 类型转换函数

本文介绍类型转换函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-184 类型转换函数

函数	描述
cast函数	将值从一种数据类型转换为另一种数据类型，并将数据类型提供给动态参数。

cast 函数

将值从一种数据类型转换为另一种数据类型，并将数据类型提供给动态参数。

在允许表达式的任何地方都允许使用铸造表达式。

语法：CAST(Expression as Datatype)

要将表达式转换到的数据类型是目标类型。要从中转换的表达式的数据类型是源类型。

表 6-185 参数说明

参数名称	描述	类型	是否必选
expression	源类型的表达式。	任意	是
Datatype	目标类型。	SQL数据类型，可选值为bigint、varchar、double precision、boolean、timestamp、decimal、integer、char、date。	是

表 6-186 索引数据类型和 SQL 数据类型的对应关系

索引的数据类型	SQL的数据类型
long	bigint
string	varchar
float	double precision

返回值类型：目标数据类型。

示例：select cast(time as date)

表 6-187 查询分析结果

类型	场景
查询语句	cast(time as date)
返回结果	2023-08-31 23:11:17.000

6.5.8 数学计算函数

本文介绍数学计算函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-188 数学计算函数

函数	描述
round函数	用于对x进行四舍五入。如果n存在，则保留n位小数；如果n不存在，则对x进行四舍五入取整数。

round 函数

用于对x进行四舍五入。如果n存在，则保留n位小数；如果n不存在，则对x进行四舍五入取整数。

- 对x进行四舍五入取整数。
  语法：ROUND(x)
- 对x进行四舍五入且保留n位小数。
  语法：ROUND(x, n)

表 6-189 参数说明

参数名称	描述	类型	是否必选
x	原始字段。	number	是
n	n位小数(int)。	int	是

返回值类型：Number

示例：select ROUND(3.1415, 1)

表 6-190 查询分析结果

类型	场景
查询语句	ROUND(3.1415, 1)
返回结果	3.1

6.5.9 URL 函数

本文介绍URL函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-191 URL 函数

函数	描述
url_encode函数	对URL进行编码。
url_decode函数	对URL进行解码。
url_extract_fragment函数	提取URL中的fragment信息。
url_extract_host函数	提取URL中的host信息。
url_extract_parameter函数	提取URL中的参数信息。
url_extract_path函数	分割目标字符串，返回被分割后的子串合集。
url_extract_port函数	提取URL中的端口信息。
url_extract_protocol函数	提取URL中的协议信息。
url_extract_query函数	提取URL中查询部分的信息。

url_encode 函数

对URL进行编码。

语法：url_encode(expr)

表 6-192 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_ENCODE('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-193 查询分析结果

类型	场景
查询语句	URL_ENCODE('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

类型	场景
返回结果	http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment

url_decode 函数

对URL进行解码。

语法：url_decode(expr)

表 6-194 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_DECODE('http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment')

表 6-195 查询分析结果

类型	场景
查询语句	URL_DECODE('http%3A%2F%2Fusername%3Apassword%40host%3A8080%2Findex%3FparameterName%3DparameterValue%23fragment')
返回结果	http://username:password@host:8080/index?parameterName=parameterValue#fragment

url_extract_fragment 函数

提取URL中的fragment信息。

语法：url_extract_fragment(expr)

表 6-196 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_FRAGMENT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-197 查询分析结果

类型	场景
查询语句	URL_EXTRACT_FRAGMENT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	fragment

url_extract_host 函数

提取URL中的host信息。

语法：url_extract_host(expr)

表 6-198 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_HOST('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-199 查询分析结果

类型	场景
查询语句	URL_EXTRACT_HOST('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	host

url_extract_parameter 函数

提取URL中的参数信息。

语法：url_extract_parameter(expr, paramName)

表 6-200 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是
paramName	参数名称	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_PARAMETER('http://username:password@host:8080/index?parameterName=parameterValue#fragment', 'parameterName')

表 6-201 查询分析结果

类型	场景
查询语句	URL_EXTRACT_PARAMETER('http://username:password@host:8080/index?parameterName=parameterValue#fragment', 'parameterName')
返回结果	parameterValue

url_extract_path 函数

提取URL中的path信息。

语法：url_extract_path(expr)

表 6-202 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：

String类型

示例：SELECT URL_EXTRACT_PATH('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-203 查询分析结果

类型	场景
查询语句	URL_EXTRACT_PATH('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	/index

url_extract_port 函数

提取URL中的端口信息。

语法：url_extract_port(expr)

表 6-204 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_PORT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-205 查询分析结果

类型	场景
查询语句	URL_EXTRACT_PORT('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	8080

url_extract_protocol 函数

提取URL中的协议信息。

语法：url_extract_protocol(expr)

表 6-206 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_PROTOCOL('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-207 查询分析结果

类型	场景
查询语句	URL_EXTRACT_PROTOCOL('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	http

url_extract_query 函数

提取URL中的查询部分的信息。

语法：url_extract_query(expr)

表 6-208 参数说明

参数名称	描述	类型	是否必选
expr	URL字符串	String	是

返回值类型：String类型

示例：SELECT URL_EXTRACT_QUERY('http://username:password@host:8080/index?parameterName=parameterValue#fragment')

表 6-209 查询分析结果

类型	场景
查询语句	URL_EXTRACT_QUERY('http://username:password@host:8080/index?parameterName=parameterValue#fragment')
返回结果	parameterName=parameterValue

6.5.10 IP 地址函数

本文介绍IP地址函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-210 IP 地址函数

函数	描述
ip_to_domain函数	判断目标IPv4地址是内网地址还是外网地址。
ip_prefix函数	获取目标IPv4地址的前缀。
is_prefix_subnet_of函数	判断目标IPv4网段是否为某网段的子网。
is_subnet_of函数	判断目标IPv4地址是否在某网段内。
ip_subnet_max函数	获取IPv4网段中的最大IP地址。
ip_subnet_min函数	获取IPv4网段中的最小IP地址。
ip_subnet_range函数	获取IPv4网段范围。
ip_to_city函数	返回城市的名称。
ip_to_provider函数	返回网络运营商的名称。

ip_to_domain 函数

判断目标IPv4地址是内网地址还是外网地址。

语法：ip_to_domain (expr)

表 6-211 参数说明

参数名称	描述	类型	是否必选
expr	IP地址	String	是

返回值类型：String类型

示例：SELECT IP_TO_DOMAIN('10.110.10.210'), IP_TO_DOMAIN('192.175.4.1')

表 6-212 查询分析结果

类型	场景1	场景2
查询语句	IP_TO_DOMAIN('10.110.10.210')	IP_TO_DOMAIN('192.175.4.1')
返回结果	intranet	internet

ip_prefix 函数

获取目标IPv4地址的前缀。

语法：ip_prefix(expr, prefixBit)

表 6-213 参数说明

参数名称	描述	类型	是否必选
expr	IPv4地址	String	是
prefixBit	前缀位数	String	是

返回值类型：String类型

示例：SELECT IP_PREFIX('10.110.10.210', 8), IP_PREFIX('144.101.32.5', 12)

表 6-214 查询分析结果

类型	场景1	场景2
查询语句	IP_PREFIX('10.110.10.210', 8)	IP_PREFIX('144.101.32.5', 12)
返回结果	10.0.0.0/8	144.96.0.0/12

is_prefix_subnet_of 函数

判断目标IPv4网段是否为某网段的子网。

语法：is_prefix_subnet_of(expr1, expr2)

表 6-215 参数说明

参数名称	描述	类型	是否必选
expr1	给定的IPv4网段	String	是
expr2	目标IPv4网段	String	是

返回值类型：Boolean类型

示例：SELECT IS_PREFIX_SUBNET_OF('192.168.0.1/24', '192.168.1.1/24')

表 6-216 查询分析结果

类型	场景
查询语句	IS_PREFIX_SUBNET_OF('192.168.0.1/24', '192.168.1.1/24')

类型	场景
返回结果	false

is_subnet_of 函数

判断目标IPv4地址是否在某网段内。

语法：is_subnet_of(expr1, expr2)

表 6-217 参数说明

参数名称	描述	类型	是否必选
expr1	给定的IPv4网段	String	是
expr2	目标IPv4地址	String	是

返回值类型：Boolean类型

示例：SELECT IS_SUBNET_OF('192.168.0.1/24', '192.168.1.1')

表 6-218 查询分析结果

类型	场景
查询语句	IS_SUBNET_OF('192.168.0.1/24', '192.168.1.1')
返回结果	false

ip_subnet_max 函数

获取IPv4网段中的最大IP地址。

语法：ip_subnet_max(expr)

表 6-219 参数说明

参数名称	描述	类型	是否必选
expr	IPv4网段	String	是

返回值类型：String类型

示例：SELECT IP_SUBNET_MAX('192.120.80.128/10')

表 6-220 查询分析结果

类型	场景
查询语句	IP_SUBNET_MAX('192.120.80.128/10')
返回结果	192.127.255.254

ip_subnet_min 函数

获取IPv4网段中的最小IP地址。

语法：ip_subnet_min (expr)

表 6-221 参数说明

参数名称	描述	类型	是否必选
expr1	IPv4网段	String	是

返回值类型：String类型

示例：SELECT IP_SUBNET_MIN('192.120.80.128/10')

表 6-222 查询分析结果

类型	场景
查询语句	IP_SUBNET_MIN('192.120.80.128/10')
返回结果	192.64.0.1

ip_subnet_range 函数

获取IPv4网段范围。

语法：ip_subnet_range(expr)

表 6-223 参数说明

参数名称	描述	类型	是否必选
expr1	IPv4网段	String	是

返回值类型：String类型

示例：SELECT IP_SUBNET_RANGE('192.120.80.128/10')

表 6-224 查询分析结果

类型	场景
查询语句	IP_SUBNET_RANGE ('192.120.80.128/10')
返回结果	["192.64.0.1","192.127.255.254"]

说明

在计算给定网段的最大IP地址、最小IP地址和IP范围时，排除了网段的单播和广播地址，计算出的IP地址为网段的最大可用IP地址、最小可用IP地址和可用IP地址范围。

ip_to_city 函数

返回城市的名称。

语法：ip_to_city(x)

表 6-225 参数说明

参数名称	描述	类型	是否必选
x	参数值为IPv4地址	String	是

返回值类型：String类型

示例：select ip_to_city(ip)

查询分析结果：

表 6-226 查询分析结果

类型	场景
查询语句	ip_to_city(ip)
返回结果	Cochise

ip_to_provider 函数

返回网络运营商的名称。

语法：ip_to_provider(x)

表 6-227 参数说明

参数名称	描述	类型	是否必选
x	参数值为IPv4地址	String	是

返回值类型：

String类型

示例：select ip_to_provider(ip)

表 6-228 查询分析结果

类型	场景
查询语句	ip_to_provider(ip)
返回结果	CONUS-YPG

6.5.11 电话号码函数

本文介绍电话号码函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-229 电话号码函数

函数	描述
mobile_carrier函数	分析电话号码所属的运营商。
mobile_city函数	分析电话号码所属的城市。
mobile_province函数	分析电话号码所属的省份。

mobile_carrier 函数

分析电话号码所属的运营商。

语法：mobile_carrier(expr)

表 6-230 参数说明

参数名称	描述	类型	是否必选
expr	电话号码	String	是

返回值类型：String类型

示例：SELECT MOBILE_CARRIER('17052294531')

表 6-231 查询分析结果

类型	场景
查询语句	MOBILE_CARRIER('17052294531')
返回结果	中国移动

mobile_city 函数

分析电话号码所属的城市。

语法：mobile_city(expr)

表 6-232 参数说明

参数名称	描述	类型	是否必选
expr	电话号码	String	是

返回值类型：String类型

示例：SELECT MOBILE_CITY('17052294531')

表 6-233 查询分析结果

类型	场景
查询语句	MOBILE_CITY('17052294531')
返回结果	西安市

mobile_province 函数

分析电话号码所属的省份。

语法：mobile_province(expr)

表 6-234 参数说明

参数名称	描述	类型	是否必选
expr	电话号码	String	是

返回值类型：String类型

示例：SELECT MOBILE_PROVINCE('17052294531')

表 6-235 查询分析结果

类型	场景
查询语句	MOBILE_PROVINCE('17052294531')
返回结果	陕西省

6.5.12 数组函数

本文介绍数组函数的语法规则，包括参数解释、函数示例等。

函数列表

表 6-236 数组函数

函数	描述
array函数	将输入的参数构造成数组，参数类型必须相同。
array_position函数	获取指定元素的下标，下标从1开始。如果指定元素不存在，则返回0。
cardinality函数	计算数组中元素的个数。
mv_length函数	计算数组中元素的个数， 同cardinality。
contains函数	判断数组中是否包含指定元素。如果包含，则返回true。
mv_contains函数	判断数组中是否包含指定元素。如果包含，则返回true，同contains。
mv_prepend函数	将指定的元素添加到数组的开始位置。
mv_append函数	将指定的元素添加到数组的末尾。
mv_slice函数	返回从start到end索引的数组。
mv_to_string函数	通过str指定分隔符连接arr所有元素。
string_to_mv函数	使用指定的分隔符str2将str1拆分为数组
mv_offset函数	返回所提供的基于0的索引处的数组元素。
mv_ordinal函数	返回所提供的基于1的索引处的数组元素。
mv_offset_of函数	返回数组中第一次出现expr的基于0的索引，如果未出现，则返回-1。
mv_ordinal_of函数	返回数组中第一次出现expr的基于1的索引，如果未出现，则返回-1。

array 函数

将参数构建成数组，参数类型必须相同。

语法：array[expr1,expr ...]

表 6-237 参数说明

参数名称	描述	类型	是否必选
expr	原始数据	String/Integer/Long/Double/Float	是

返回值类型：Array类型

示例：SELECT ARRAY['1','2','3','4','5']

表 6-238 查询分析结果

类型	场景
查询语句	ARRAY['1','2','3','4','5']
返回结果	["1", "2", "3", "4", "5"]

array_position 函数

获取指定元素的下标，下标从1开始。如果指定元素不存在，则返回0。

语法：array_position(expr, ele)

表 6-239 参数说明

参数名称	描述	类型	是否必选
expr	原始数组	数组类型。	是
ele	指定的元素	数组中的一个元素，必须与数组中元素类型相同。	是

返回值类型：Integer类型

示例：SELECT ARRAY_POSITION(ARRAY['1','2','3'],'2')

表 6-240 查询分析结果

类型	场景
查询语句	ARRAY_POSITION(ARRAY['1','2','3'],'2')
返回结果	2

cardinality 函数

计算数组中元素的个数，参数类型必须相同。

语法：cardinality(expr)

表 6-241 参数说明

参数名称	描述	类型	是否必选
expr	原始数组	Array（String/Number）	是

返回值类型：Integer类型

示例：SELECT CARDINALITY(ARRAY['1','2','3'])

表 6-242 查询分析结果

类型	场景
查询语句	CARDINALITY(ARRAY['1','2','3'])
返回结果	3

mv_length 函数

计算数组中元素的个数， 同cardinality。

语法：mv_length(expr)

表 6-243 参数说明

参数名称	描述	类型	是否必选
expr	原始数组	Array（String/Number）	是

返回值类型：Integer类型

示例：SELECT MV_LENGTH(ARRAY['1','2','3'])

表 6-244 查询分析结果

类型	场景
查询语句	MV_LENGTH (ARRAY['1','2','3'])
返回结果	3

contains 函数

判断数组中是否包含指定元素。如果包含，则返回true。

语法：contains(expr, ele)

表 6-245 参数说明

参数名称	描述	类型	是否必选
expr	原始数组	Array（String/Number）	是
ele	指定的元素	String/Number，必须与数组中元素类型相同。	是

返回值类型：Boolean类型

示例：SELECT CONTAINS(ARRAY['1','2'],'1')

表 6-246 查询分析结果

类型	场景
查询语句	CONTAINS(ARRAY['1','2'],'1')
返回结果	true

mv_contains 函数

判断数组中是否包含指定元素。如果包含，则返回true，同contains。

语法：mv_contains(expr, ele)

表 6-247 参数说明

参数名称	描述	类型	是否必选
expr	原始数组	Array（String/Number）	是
ele	指定的元素	String/Number，必须与数组中元素类型相同。	是

返回值类型：Boolean类型

示例：SELECT MV_CONTAINS(ARRAY['1','2'],'1')

表 6-248 查询分析结果

类型	场景
查询语句	MV_CONTAINS(ARRAY['1','2'],'1')
返回结果	true

mv_prepend 函数

将指定的元素添加到数组的开始位置。

语法：mv_prepend(expr, arr)

表 6-249 参数说明

参数名称	描述	类型	是否必选
expr	指定的元素	String/Number，必须与数组中元素类型相同。	是
arr	原始数组	Array（String/Number）	是

返回值类型：Array类型

示例：SELECT MV_PREPEND('1', ARRAY ['1','2'])

表 6-250 查询分析结果

类型	场景
查询语句	MV_PREPEND ('1', ARRAY['1','2'])
返回结果	["1","1","2"]

mv_append 函数

将指定的元素添加到数组的末尾。

语法：mv_append(arr, expr)

表 6-251 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
expr	指定的元素	String/Number，必须与数组中元素类型相同。	是

返回值类型：Array类型

示例：SELECT MV_APPEND(ARRAY['1','2'],'1')

表 6-252 查询分析结果

类型	场景
查询语句	MV_APPEND(ARRAY['1','2'], '1')
返回结果	["1","2","1"]

mv_slice 函数

返回从start到end索引的数组。

语法：mv_slice(arr, start, end)

表 6-253 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
start	起始位置	Integer	是
end	结束位置	Integer	是

返回值类型：Array类型

示例：SELECT MV_SLICE(ARRAY['1','2','3','4','5'], 2, 4)

表 6-254 查询分析结果

类型	场景
查询语句	MV_SLICE(ARRAY['1','2','3','4','5'], 2, 4)
返回结果	["3","4"]

mv_to_string 函数

使用指定的分隔符str连接arr所有元素。

语法：mv_to_string(arr, str)

表 6-255 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
str	指定字符	String	是

返回值类型：String类型

示例：SELECT MV_TO_STRING(ARRAY['1','2','3','4','5'],'-')

表 6-256 查询分析结果

类型	场景
查询语句	MV_TO_STRING(ARRAY['1','2','3','4','5'],'-')

类型	场景
返回结果	1-2-3-4-5

string_to_mv 函数

使用指定的分隔符str2将str1拆分为数组。

语法：string_to_mv(str1, str2)

表 6-257 参数说明

参数名称	描述	类型	是否必选
str1	原始字符串	String	是
str2	指定字符	String	是

返回值类型：Array类型

示例：SELECT STRING_TO_MV('1-2-3-4-5','-')

表 6-258 查询分析结果

类型	场景
查询语句	STRING_TO_MV('1-2-3-4-5','-')
返回结果	["1","2","3","4","5"]

mv_offset 函数

返回所提供的基于0的索引处的数组元素，或对于超出范围的索引返回null。

语法：mv_offset(arr, index)

表 6-259 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
index	指定索引位置	Integer	是

返回值类型：String/Integer/Long/Boolean/Double类型

示例：SELECT MV_OFFSET(ARRAY['1','2','3','4','5'], 2)

表 6-260 查询分析结果

类型	场景
查询语句	MV_OFFSET(ARRAY['1','2','3','4','5'], 2)
返回结果	3

mv_ordinal 函数

返回所提供的基于1的索引处的数组元素，或对于超出范围的索引返回null。

语法：mv_ordinal(arr, index)

表 6-261 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
index	指定索引位置	Integer	是

返回值类型：String/Integer/Long/Boolean/Double类型

示例：SELECT MV_ORDINAL(ARRAY['1','2','3','4','5'], 2)

表 6-262 查询分析结果

类型	场景
查询语句	MV_ORDINAL (ARRAY['1','2','3','4','5'], 2)
返回结果	2

mv_offset_of 函数

返回数组中第一次出现expr的基于0的索引，如果未出现，则返回-1。

语法：mv_offset_of(arr, expr)

表 6-263 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是

参数名称	描述	类型	是否必选
expr	指定元素	String/Number，必须与数组中元素类型相同。	是

返回值类型：Integer类型

示例：SELECT MV_OFFSET_OF(ARRAY['1','2','3','4','5'], '2')

表 6-264 查询分析结果

类型	场景
查询语句	MV_OFFSET_OF(ARRAY['1','2','3','4','5'], '2')
返回结果	1

mv_ordinal_of 函数

返回数组中第一次出现expr的基于1的索引，如果未出现，则返回-1。

语法：mv_ordinal_of(arr, expr)

表 6-265 参数说明

参数名称	描述	类型	是否必选
arr	原始数组	Array（String/Number）	是
expr	指定元素	String/Number，必须与数组中元素类型相同。	是

返回值类型：Integer类型

示例：SELECT MV_ORDINAL_OF(ARRAY['1','2','3','4','5'], '2')

表 6-266 查询分析结果

类型	场景
查询语句	MV_ORDINAL_OF(ARRAY['1','2','3','4','5'], '2')

类型	场景
返回结果	2

7 日志可视化

7.1 日志可视化概述

云日志服务支持可视化展示您的查询与分析结果。您可以根据分析需求选用合适的统计图表类型或仪表盘展示查询和分析结果，并将结果保存到实时刷新的仪表盘中。

表 7-1 可视化方式

可视化方式	说明
统计图表	统计图表是云日志服务根据**SQL查询语法**渲染出的结果。云日志服务提供表格、线图、柱状图等多种图表类型。详细请参考**使用统计图表将日志可视化**。
仪表盘	• 仪表盘是云日志服务提供的实时数据分析大盘。您可以在仪表盘查看多个基于SQL语句查询分析结果的统计图表，并能将多张统计图表同步保存到仪表盘中。详细请参考**使用仪表盘将日志可视化**。 • 云日志服务提供多种仪表盘模板，详细请参考**日志仪表盘模板**。

7.2 使用统计图表将日志可视化

7.2.1 统计图表概述

云日志服务LTS支持通过统计图表的方式对查询和分析的结果进行可视化展示。用户还可以将图表分析结果保存到仪表盘，进行长期监控。

- 统计图表用于展示查询和分析结果，支持在图表栏中选择不同图表，并进行相应设置。
- 支持对统计图表进行全局配置。例如选择颜色方案后，整个统计图表的颜色都基于该颜色方案进行展示。

- 在统计图表页面，选择合适的图表类型。

限制说明

说明

目前此功能仅支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布二零一、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华南-深圳，支持部分白名单用户使用的局点有：亚太-曼谷、华北-北京一、华东-上海二、华北-乌兰察布二零二，其他局点暂不支持该功能。

规格与限制如下：

- 一个日志流最多可创建100个图表。
- 一个仪表盘最多可添加50个图表。

统计图表类型

支持对图表进行新建、保存、另存为等操作。详细请参考表7-2。

支持使用表格、柱状图、折线图等图表类型展示不同场景数据，详细请参考表7-3。

表 7-2 操作说明

功能名称	功能描述
新建	选择不同的图表类型，将图表分析结果保存到仪表盘。
保存	对当前可视化图表进行保存。
另存为	对已有可视化图表进行复制。
下载	将图表分析结果下载到excel。
展开图表	可展开当前日志流下的可视化图表。
收起图表	可收起当前日志流下的可视化图表。

表 7-3 图表类型

图表类型	使用场景
表格	表格是最常见的数据展示类型，通过对数据结构化的整理，实现数据的对比与统计。大多数场景均适用。
柱状图	柱状图描述的是分类数据，直观表现每一个分类项的大小对比关系。统计近一天各错误码类型出现的次数等分类统计场景适用。
折线图	折线图需要统计数据具备时序字段，依据时间顺序组织与聚合指标。可直观反映指标随时间的变化趋势。
饼图	饼图描述的是不同分类的占比情况，通过扇区大小来衡量各分类项的占比情况。错误码占比情况分析等占比统计场景适用。

图表类型	使用场景
数字图	数字图描述的是单个指标，一般选择具备有业务价值的关键性指标。统计天、周、月 PV、UV 等单指标场景适用。
数字折线图	折线图和数字图的组合。折线图用于表示数据趋势和变化的，数字图则展示关键性指标。在一些需要同时显示趋势和关键数据点的场合适用。
地图	地图通过图形的位置来表现数据的地理位置，通常来展示数据在不同地理区域上的分布情况。攻击 IP 地理分布等地理位置统计场景适用。
漏斗图	漏斗图适用于单流向单路径的业务流程，对各环节进行统计并用梯形面积表示某个环节业务量与上一个环节之间的差异。

7.2.2 LTS 表格

表格作为最常见的数据展示类型，是组织整理数据最基本的手段，通过对数据的整理，达到快速引用和分析的目的。通过查询分析语法得到的数据结果默认以表格方式进行展示。

查看表格

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 配置表格参数，查看表格。

表 7-4 配置参数

类别	参数	说明
标准配置	格式化	将表格数据按照指定格式进行显示。
	单位	自定义配置表格数据的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。
查询分析设置	隐藏字段	选择目标字段，将该字段在表格中隐藏。
表格配置	每页显示	每页显示的数据条数。
	显示总数	显示表格数据的总条目数。

类别	参数	说明
列配置	对齐方式	表格数据的对齐方式，支持左对齐，右对齐以及居中。
	开启搜索	开启后，即可对表格列数据进行搜索功能。
	开启排序	开启后，即可对表格列数据进行排序功能。
	字体大小	表格字体的大小，取值范围为12px~24px 。

----结束

7.2.3 LTS 柱状图

柱状图是使用垂直或水平的柱子显示类别之间的数值比较，用于描述分类数据，并统计每一个分类中的数量。

云日志服务（LTS）提供的柱状图，有垂直柱子和水平柱子，矩形块宽度一定，高度代表数值大小。有多列数据映射到Y轴时，采用分组柱状形式显示。

默认采用垂直柱子，您可以根据自己的需要进行选择。基本构成如下：

- X轴（横轴）
- Y轴（纵轴）
- 矩形块
- 图例

查看柱状图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 在通用配置页签中，配置柱状图参数，查看柱状图。

表 7-5 柱状图参数说明表

类别	参数	说明
标准配置	格式化	将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。

类别	参数	说明
	图表名称字号	设置图表名称的字号大小。
柱配置	方向	选择基础柱状图或横向柱状图。
	柱宽度	设置柱宽度。
	是否显示值	开启后，显示各个条形体对应的数值。
	值字体大小	设置各个条形体对应的数值字体大小
	是否堆叠	开启后，将堆叠显示Y轴数据。
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
图例配置	隐藏图例	开启后，可以隐藏图例和对比值的显示。
	图例位置	图例在图表中的位置，选择图表顶部或图表右边。
	对比数值	选择显示最大值、最小值、平均值、求和值等，可勾选多个。
图形配置	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。
Tooltip配置	不排序、升序、降序	提示框配置，当Y轴数据选择多个时，可对其进行排序显示。
X轴	显示X轴	开启后，显示X轴数据。
	X轴名称	设置X轴名称。
Y轴	显示Y轴	开启后，显示Y轴数据。
	Y轴名称	设置Y轴名称。
	Y轴位置	设置Y轴位置，左边或者右边。

----结束

7.2.4 LTS 折线图

线图属于趋势类分析图表，一般用于表示一组数据在一个有序数据类别（多为连续时间间隔）上的变化情况，用于直观分析数据变化趋势。在线图中，可以清晰的观测到数据在某一个周期内的变化，主要反映在：

- 递增性或递减性
- 增减的速率情况

- 增减的规律（如周期变化）
- 峰值和谷值

所以，线图是用于分析数据随时间变化趋势的最佳选择。同时，也可以绘制多条线用于分析多组数据在同一时间周期的变化趋势，进而分析数据之间的相互作用和影响（如同增同减，成反比等）。

查看折线图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 在通用配置页签中，配置折线图参数，查看折线图。

表 7-6 折线图参数说明表

类别	参数	说明
标准配置	格式化	将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
	维度列	请从下拉列表中选择，一般为有序数据类别。
	趋势对比	当X轴为时间数据时，且不设置维度列时，可开启该按钮。 开启后，设置比较对象时间，时间小于等于24小时。设置完成后，将当前时间的数据与对象时间数据进行比较。
图例配置	隐藏图例	开启后，可以隐藏图例和对比值的显示。
	图例位置	选择图表顶部或图表右边。
	对比数值	选择显示最大值、最小值、平均值、求和值等，可勾选多个。
图形配置	连接方式	设置线图显示格式，可选择直线或曲线。
	线宽	折线的线宽。

类别	参数	说明
	是否显示点	开启该功能后，显示折线的连接点。
	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。
Tooltip配置	排序方式	提示框配置，当Y轴数据选择多个时，可对其进行排序显示。
X轴	显示X轴	开启后，显示X轴数据。
	X轴名称	设置X轴名称。
Y轴	显示Y轴	开启后，显示Y轴数据。
	Y轴名称	设置Y轴名称。
	Y轴位置	设置Y轴位置，左边或者右边。

----结束

7.2.5 LTS 饼图

饼图用于表示不同分类的占比情况，通过弧度大小来对比各种分类。饼图通过将一个圆饼按照分类的占比划分成多个区块，整个圆饼代表数据的总量，每个区块表示该分类占总体的比例大小，所有区块的加和等于100%。基本构成如下：

- 扇形
- 文本百分比
- 图例

类型

云日志服务（LTS）提供默认饼图、环图及南丁格尔玫瑰图三种类型的饼图。

- 环图

  环图本质上是将饼图中心挖空，相比于饼图来说有如下优点：

  - 在原有构成的基础上增加了总数显示，展示了更多的信息。
  - 两个饼图直接进行比较是非常不直观的，两个环图间可以通过环状条长度进行简单的对比。

- 南丁格尔玫瑰图

  南丁格尔玫瑰图本质上并不是环图，而是在极坐标系下画出来的柱状图，每一个分类数据被圆弧平分，使用圆弧的半径长短表示数据的大小，相比于饼图来说有如下优点：

  - 饼图适用于不超过10条的分类数据，南丁格尔玫瑰图则适用于分类较多的场景（10-30条数据）。

- 由于半径和面积是成平方的关系，南丁格尔玫瑰图放大了各个分类数据之间值的差异，尤其适合对比大小相近的数值。
- 由于圆形有周期的特性，南丁格尔玫瑰图也适用于表示一个周期的时间概念，比如星期、月份。

查看饼图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 配置饼图参数，查看饼图。

类别	参数	说明
标准配置	格式化	将Y轴按照指定格式进行显示。
	单位	自定义配置Y轴的单位。
	小数点位数	设置显示数值小数点位数。
	图表名称字号	设置图表名称的字号大小。
饼图配置	饼图类型	包括饼图、环图和南丁格尔玫瑰图。
	是否显示刻度	开启后，显示饼图图形上的文本标签，可用于说明图形的一些数据信息，比如值，名称等。
	刻度文本格式	可配置为分类、百分比、分类：百分比或分类：数值（百分比）。
	标签位置	开启是否显示刻度后，可配置此参数，调整标签在图表中的位置。
查询分析设置	数据	分类数据对应的数值。
	第一层数据	
	类目	分类数据。
	展示数量	显示分类数据的个数。
	排序方式	升序或降序。
	其余归为其他	开启后，除了展示的数据，其余归为其他方式展示。
	添加分层	单击添加分层，设置第二层数据，每层数据包括类目、展示数量、排序方式、其余归为其他。
图例配置	隐藏图例	开启后，可以隐藏图例和图例内容的显示。

类别	参数	说明
	图例内容	选择显示值和百分比，可勾选多个。
	图例位置	图例在图表中的位置，选择图表顶部或图表右边。
图形配置	外半径	指定饼图外半径值。取值范围为40~100 。
	内半径	指定饼图内半径值。取值范围为0~100。
	上边距	坐标轴距离图表上边界距离。
	下边距	坐标轴距离图表下边界距离。
	左边距	坐标轴距离图表左边界距离。
	右边距	坐标轴距离图表右边界距离。

----结束

7.2.6 LTS 数字图

数字图通常用来表示单一数据点或关键性指标，能够更好地展示单一信息和数据的相对大小，是一种非常清晰的信息展示方式，适用于需要重点突出关键信息和数据的场合。通过数字图可以快速地呈现信息和数据，使得用户能够快速、直观地理解数据的趋势和关键指标。

查看数字图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 配置数字图参数，查看数字图。

类别	参数	说明
查询分析设置	数值列	支持数字或字符串数据。
	同比数据	选择待对比的字段，在图表中显示该字段对应的值。
主体设置	图表名称字号	设置图表名称的字号大小。
	格式化	将数据按照指定格式进行显示。
	数值字号	显示值的字号，取值范围为12px~80px。
	数值单位	显示值的单位

类别	参数	说明
	单位字号	显示值单位的字号，取值范围为12px~50px。
	小数点位数	设置显示数值小数点位数。
	添加对比值	开启后，显示待对比字段对应的值。
	对比值格式化	将待对比数据按照指定格式进行显示。
	对比值字号	待对比值的字号，取值范围为12px~50px。
	对比值单位	待对比值的单位。
	对比值单位字号	显示待对比值单位的字号，取值范围为12px~50px。
	描述	对显示的数值及对比值趋势的描述，显示在数值下方。
背景配置	背景色	图表的背景颜色，支持深色或浅色。

----结束

7.2.7 LTS 数字折线图

数字图和折线图组合，同时显示趋势和关键数据点。可以帮助用户更好的理解数据和趋势变化，从而更好的进行业务决策。

查看数字折线图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 配置数字折线图参数，查看数字折线图。

类别	参数	说明
查询分析设置	X轴数据	支持数字或字符串数据。
	Y轴数据	支持数字或字符串数据，可以选择多个数据。
图表样式	连接方式	设置线图显示格式，可选择直线或曲线。
主体设置	图表名称字号	设置图表名称的字号大小。
	数据格式	将数据按照指定格式进行显示。

类别	参数	说明
	数值字号	显示值的字号，取值范围为12px~80px。
	数值单位	显示值的单位。
	单位字号	显示值单位的字号，取值范围为12px~50px。
	小数点位数	设置显示数值小数点位数。
背景配置	背景色	图表的背景颜色，支持深色或浅色。

----结束

7.2.8 LTS 地图

以地图作为背景，通过图形颜色、图像标记的方式展示地理数据信息。云日志服务（LTS）提供的地图，包括中国地图和世界地图。您在查询和分析语句中使用特定函数（中国地图：ip_to_province函数，世界地图：ip_to_country函数）后，云日志服务（LTS）将以地图形式展示分析结果。

基本构成如下：

- 地图画布
- 色块

查看地图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击  图标。

步骤6 配置地图参数，查看地图。

表 7-7 地图参数说明表

参数	说明
地图类型	区域定位：设置地图显示的区域范围，包含中国省级地图和世界地图两种。
省份	地图类型为中国省级地图，该字段为具体省份。例如浙江省
国家	地图类型为世界地图，该字段为具体国家。例如中国
数值列	选择用于展示数值的字段。
图表名称字号	设置图表名称的字号大小。

----结束

7.2.9 LTS 漏斗图

漏斗图适用于业务流程比较规范、周期长、环节多的单流程单向分析，通过漏斗各环节业务数据的比较能够直观地发现和说明问题所在的环节，进而做出决策。漏斗图用梯形面积表示某个环节业务量与上一个环节之间的差异。

查看漏斗图

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中，选择“日志管理”，进入日志管理页面。

步骤3 在日志管理页面中，选择目标日志组和日志流，进入日志流详情页面。

步骤4 在日志流详情页面中，输入查询和分析语句，然后单击15分钟（相对），设置查询和分析的时间范围。

步骤5 选择“日志分析”，单击 图标。

步骤6 配置漏斗图参数，查看漏斗图。

表 7-8 漏斗图参数说明表

参数	说明
系列名称	漏斗图的名称。
数值列	选择数值字段，某个字段对应的数值越大，在越上面。
隐藏图例	开启后，可以隐藏漏斗图上方的字段名显示。

----结束

7.3 使用仪表盘将日志可视化

7.3.1 创建日志仪表盘

仪表盘能够在云日志服务中实时分析日志数据，SQL语句查询分析出的日志结果可由多种图表表示，并能将多张统计图表同步保存到仪表盘中。

说明

目前此功能仅支持全部用户使用的局点有：华南-广州、华北-北京四、华北-乌兰察布二零一、华北-乌兰察布一、华东-上海一、中国-香港、西南-贵阳一、亚太-新加坡、华南-深圳，支持部分白名单用户使用的局点有：亚太-曼谷、华北-北京一、华东-上海二、华北-乌兰察布二零二，其他局点暂不支持该功能。

前提条件

- 已成功采集到日志。
- 对日志内容已完成结构化配置，具体操作请参考**结构化配置**。

限制条件

- 一个账号最多可创建100个仪表盘。
- 一个日志流最多可创建100个图表。
- 一个仪表盘最多可添加50个图表。

创建仪表盘

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘 ”。

步骤2 单击  ，填写分组名称，添加仪表盘分组。

步骤3 单击确定，创建分组成功后，单击“添加仪表盘”，进入仪表盘创建页面。

步骤4 在仪表盘创建页面，配置创建仪表盘相关参数。

表 7-9 创建仪表盘参数

参数名称	说明
仪表盘名称	仪表盘的名称。仅支持中英文、数字、中划线、下划线、小数点，不能以小数点开头和结尾，长度不超过255。
企业项目	选择业务需要的企业项目，也可单击“查看企业项目”，在企业项目管理页面查看全部企业项目。
添加到仪表盘分组	若没有选中分组就添加仪表盘，支持开启该按钮，显示**分组类型**： ● 已有分组：选择已有的仪表盘分组。 ● 新建分组：输入仪表盘分组名称。 若已选中分组再添加仪表盘，则默认开启该按钮且显示分组类型为已有分组。
简洁模式	简洁模式打开后，仪表盘界面不再显示编辑、删除、添加过滤器、图表下钻等按钮。
添加图表	● 添加可视化图表：将日志流的可视化图表加入仪表盘 ● 使用仪表盘模板：有自定义模板和系统模板两种类型： – 自定义模板：用户从已创建的仪表盘中提取的模板。 – 系统模板：系统账号定义的模板，用户无法修改。

步骤5 添加图表。

- 添加可视化图表

  在“添加图表”区域中，鼠标悬浮在添加可视化图表模块，单击“开始添加图表”，进入添加可视化图表界面。选择业务需要的日志流，根据业务需要勾选一

个或多个图表名称前的，单击“确定”，进入仪表盘详情页后，调整图表信息，单击“保存设计”。

如果当前日志流未配置或没有当前需要的可视化图表，单击“前往添加图表”，新建图表。

- 使用仪表盘模板

  在“添加图表”区域中，鼠标悬浮在使用仪表盘模板模块，单击“使用仪表盘模板”，进入使用仪表盘模板界面。根据业务需要选择仪表盘模板，单击下一步选择业务需要的日志流，根据业务需要勾选一个或多个日志流名称前的，单击“确定”。

步骤6 仪表盘创建成功后，在仪表盘列表生成一条仪表盘信息。

说明

- 单击仪表盘操作列的编辑，修改仪表盘名称和简洁模式。
- 单击仪表盘操作列的移动分组，修改仪表盘分组。
- 单击仪表盘操作列的删除按钮即可删除删除仪表盘。

----结束

新建可视化图表到仪表盘

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘目录下方，选中仪表盘分组，单击待操作的仪表盘名称进入详情页。

步骤3 单击，在添加可视化图表界面中，选择相应日志流。单击“前往添加图表”。

步骤4 参照表7-10填写相关参数，填写完成后单击“确定”。

表 7-10 创建图表

参数	说明
图表名称	用于区分日志流下不同的图表。 • 仅支持中英文、数字、中划线、下划线、空格、括号、小数点，不能以小数点、空格开头或结尾。 • 长度为1-64个字符。
可视化对象	• 默认语句“SELECT * ”，表示查询该日志流内的结构化数据，其中*为结构化字段。 • 如需自行编辑SQL语句，请参考设置云端结构化解析日志相关内容。
图表类型	LTS提供多种图表类型供用户选择：表格、柱状图、折线图等。

参数	说明
同时添加到仪表盘	• 单击“同时添加到仪表盘”后的，勾选一个或多个仪表盘前面的，可将图表同步添加至仪表盘。 • 如果关闭“同时添加到仪表盘”后的，则表示新建图表不在仪表盘显示。

步骤5 在图表编辑页面，根据业务需求参考**分析LTS日志**填写相关参数，单击“确定”，新建图表在仪表盘显示成功。

----结束

相关操作

创建仪表盘后，单击仪表盘名称进入详情页，您可以对仪表盘进行如下操作。

表 7-11 相关操作

操作	说明
编辑仪表盘中的图表	将光标移至图表框右上角，单击，在下拉框中选择“编辑图表”，在可视化页面编辑图表，具体操作请参考**可视化**。
移除仪表盘中的图表	将光标移至图表框右上角，单击，在下拉框中选择“移除图表”，单击“保存设计”，可将已创建图表删除。
调整仪表盘中图表的位置	将光标移至待操作的图表框内，选中该图表，可将该图表移动至仪表盘内任意位置，单击“保存设计”，调整当前图表布局。
调整仪表盘中图表的大小	将光标移至待操作的图表框右下角边缘，选中该图表，可根据业务展示内容需求调整图表大小，单击“保存设计”，调整当前图表布局。
编辑仪表盘中的过滤器	将光标移至过滤器框右上角，单击，在下拉框中选择“编辑”，在添加过滤器页面编辑过滤器，具体操作请参考**添加过滤器**。
复制仪表盘中的过滤器	将光标移至过滤器框右上角，单击，在下拉框中选择“复制”，跳转到添加过滤器页面，单击“确定”即可复制过滤器。
删除仪表盘中的过滤器	将光标移至过滤器框右上角，单击，在下拉框中选择“删除”，在弹出的**删除过滤器**提示框中，单击“确定”即可删除过滤器。
调整仪表盘中过滤器的大小	将光标移至待操作的过滤器右下角边缘，可根据业务展示内容需求调整过滤器大小，单击“保存设计”，调整当前过滤器布局。

操作	说明
自动刷新	单击右上角的 ，开启仪表盘自动刷新功能，选择自动刷新的时间，可使仪表盘中的所有图表数据自动进行刷新。自动刷新的时间有1分钟、5分钟、15分钟。
手动刷新	选择待操作的仪表盘，单击 可手动刷新当前页面。
全屏显示	选择待操作的仪表盘，单击 ，可全屏显示仪表盘。全屏后，勾选保持在线按钮，可以保持在线状态，会话一直有效，当前账号不会退出。
退出全屏显示	将光标移至屏幕上方，单击弹出的 ，或者单击 ，或者按键盘中的“Esc”可退出全屏模式。
全屏显示单个图表	选择待操作的仪表盘，单击取消退出编辑模式。将光标移至图表框右上角，单击 ，在下拉框中选择“全屏”，可全屏显示图表数据。
退出全屏显示单个图表	将光标移至屏幕上方，单击弹出的 ，或者单击 ，在下拉框中选择“退出全屏”，或者按键盘中的“Esc”可退出全屏模式。
手动刷新单个图表	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“刷新”，或者在全屏模式下，单击 ，在下拉框中选择“刷新”，可手动刷新当前图表页面。
查询时间设置	选择待操作的仪表盘，单击 前面的下拉框 1小时(相对) 。 时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。 **说明** • 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。 • 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。 • 自定义：表示查询指定时间范围的日志数据。
查看图表详情	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“查看图表详情”，可查看图表详情。
添加告警	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“添加告警”，可新建告警规则。

操作	说明
复制	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“复制”，可复制图表到当前仪表盘。
复制到其他仪表盘	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“复制到其他仪表盘”，可将该图表复制到其他仪表盘。
复制搜索分析语句	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“复制搜索分析语句”，可复制该图表的搜索分析语句。
导出图表数据	选择待操作的仪表盘，将光标移至图表框右上角，单击 ，在下拉框中选择“导出图表数据”，可导出图表数据。

7.3.2 添加日志仪表盘过滤器

在云日志服务仪表盘中添加过滤器，即对整个仪表盘进行查询过滤或变量替换操作。

前提条件

- 已成功采集到日志。
- 已添加图表到仪表盘。
- 日志流已配置结构化规则。

限制条件

一个仪表盘最多可添加10个过滤器。

添加过滤器

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”，选择待操作的仪表盘。

步骤2 单击仪表盘名称进入详情页。

步骤3 单击 ，在过滤器页面中，配置添加过滤器的相关参数。

说明

不支持过滤数值型的字段。

表 7-12 添加过滤器参数

参数	说明
过滤器名称	过滤器名称。仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。
查询方式	过滤器选中的条件与已有图表查询条件组合方式。有AND和NOT两种方式，默认为AND。
Key值	配置需要过滤的字段。必填项，仅支持输入英文、数字、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾，且不能是纯数字。
别名	Key值的别名。仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点开头、下划线开头或以小数点结尾。
静态列表项	设置Key值对应的Value。多次单击 + 可添加多个Value。 单击 + 可添加Value，需配置如下参数： ● 列表项名称：需要过滤的Key值对应的Value字段名称。仅支持输入英文、数字、中文、中划线、下划线及小数点，且不能以小数点、下划线开头或以小数点结尾。 ● 别名：Value的别名。 ● 默认选中：开启默认选中开关，可直接对添加的Value进行过滤选中。 ● 操作：单击 删除添加的Value。
添加动态列表项	开启**添加动态列表项**的开关，可添加动态列表项，即为Key值配置动态Value。 开启**添加动态列表项**的开关，需配置如下参数： ● 日志组：选择待查询的日志组。 ● 日志流：选择待查询的日志流。 ● SQL引擎：支持管道符版本和非管道符版本。管道符版本的SQL查询默认语法为*** | select ***，非管道符版本的SQL查询默认语法为**select *** ● 动态列表来源：有两种方式分别是**字段模糊匹配**和**SQL查询**。 字段模糊匹配：选择当前日志流中配置的结构化字段。 SQL查询：输入SQL查询语句。单击“查询”，可预览动态列表项。

步骤4 完成后，单击“确定”。

----结束

7.3.3 日志仪表盘模板

7.3.3.1 仪表盘模板概述

云日志服务提供内置的仪表盘模板，有**APIG仪表盘模板**、**CCE仪表盘模板**、**CDN仪表盘模板**、**CFW仪表盘模板**、**CSE仪表盘模板**、**DCS仪表盘模板**、**DDS仪表盘模板**、**DMS仪表盘模板**、**DSL仪表盘模板**、**ELB仪表盘模板**、**ER仪表盘模板**、**METRIC仪表盘模板**、**NGINX仪表盘模板**、**VPC仪表盘模板**、**WAF仪表盘模板**、**采集诊断仪表盘模板**。

更多仪表盘模板内容请参考**日志仪表盘模板**。

说明

内置的仪表盘模板不能修改。

APIG 仪表盘模板

当日志流标签为log_type=apig_access或结构化配置了APIG模板时，可使用APIG仪表盘模板查看指标。APIG仪表盘模板分组里有三种仪表盘模板，分别是APIG监控中心、APIG访问中心和APIG秒级监控。更多内容请参考**APIG仪表盘模板**。

- **APIG监控中心**：该仪表盘支持通过请求域名或app_id过滤信息。主要展示APIG日志的访问量PV、访问量UV、流量、访问失败率、延迟、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP等指标。
- **APIG访问中心**：该仪表盘支持通过请求域名或app_id过滤信息。主要展示APIG日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、访问量UV分布（中国）、访问量UV分布（世界）、平均时延分布（中国）、平均时延分布（世界）、今日PV/UV、7日PV/UV、区域访问TOP10(省份)、区域访问TOP10(城市)、Host访问TOP10、UserAgent访问TOP10、设备占比(终端)、设备占比(系统)、TOP URL、TOP 访问IP等指标。
- **APIG秒级监控**：该仪表盘支持通过请求域名或app_id过滤信息。主要展示APIG日志的QPS、成功率、延迟、流量、状态码、Upstream状态码等指标。

CCE 仪表盘模板

CCE仪表盘模板分组里有7种仪表盘模板，分别是CCE日志节点操作、CCE日志K8s对象操作、CCE日志K8s事件查询、CCE日志K8s事件中心、CCE日志聚合检索、CCE日志账号操作审计、CCE日志审计中心。更多内容请参考**CCE仪表盘模板**。

- **CCE日志节点操作**：该仪表盘支持通过节点名称、操作用户、状态码或操作类型过滤信息。主要展示节点数趋势、非系统用户操作趋势、create操作状态码分布、delete操作状态码分布等。
- **CCE日志K8s对象操作**：该仪表盘支持通过命名空间、操作用户、状态码等过滤信息。主要展示重要操作趋势、非系统用户操作趋势、create操作资源类型分布、delete操作资源类型分布等。

- **CCE日志K8s事件查询：**该仪表盘支持通过事件等级、事件类型、集群ID等过滤信息。主要展示事件总数、普通事件数、重要事件分布、警告事件数等。
- **CCE日志K8s事件中心：**该仪表盘支持通过事件等级、事件类型、集群ID等过滤信息。主要展示节点FD不足、节点磁盘空间不足、事件同步异常、事件分布等。
- **CCE日志聚合检索：**该仪表盘支持通过命名空间、操作用户、状态码等过滤信息。主要展示用户分布趋势、命名空间分布趋势、操作类型分布趋势、状态码分布趋势等。
- **CCE日志账号操作审计：**该仪表盘支持通过用户名、命名空间、状态码过滤信息。主要展示资源创建数、资源修改数、资源删除数、操作命名空间分布等。
- **CCE日志审计中心：**该仪表盘支持通过命名空间、操作用户、状态码等过滤信息。主要展示总审计记录数、操作用户数、活跃节点数、异常访问次数等。

CDN 仪表盘模板

当日志流标签为log_type=cdn_access或结构化配置了CDN模板时，可使用CDN仪表盘模板查看指标。CDN仪表盘模板分组里有四种仪表盘模板，分别是CDN基础数据、CDN热门资源、CDN用户分析和CDN错误分析。更多内容请参考**CDN仪表盘模板**。

- **CDN基础数据**：该仪表盘主要展示CDN日志的缓存命中率、访问状态、访问延时分布、请求带宽、下载速度、访问次数/人数、请求命中率和访问平均延时等指标。
- **CDN热门资源：**该仪表盘主要展示CDN日志的域名访问次数Top5、域名下载流量Top5、热门访问（URI）、热门访问（来源）、全国访问次数分布统计、全国下载网速统计、省份统计、运营商流量和速度和运营商统计等指标。
- **CDN用户分析**：该仪表盘主要展示CDN日志的访问次数、访问客户端统计、运营商次数统计、访问人数、访问地区分布、有效访问用户TOP和下载量TOP用户等指标。
- CDN错误分析：该仪表盘主要展示CDN日志的错误域名访问Top5、错误URI访问Top5、错误请求状态分布、错误按运营商统计、错误按客户端统计、错误按省份统计、4XX错误详情、5XX错误详情和错误按国家统计等指标。

CFW 仪表盘模板

当日志流标签为log_type=cfw_flow或结构化配置了CFW_FLOW模板时，可使用CFW仪表盘模板查看指标。CFW仪表盘模板分组里有三种仪表盘模板，分别是CFW流量日志中心、CFW访问日志中心和CFW攻击日志中心。更多内容请参考**CFW仪表盘模板**。

- **CFW流量日志中心**：该仪表盘主要展示CFW日志的互联网访问流量趋势、互联网访问流入地域分布、互联网访问应用分布、互联网访问源IP TOP5、互联网访问目的IP TOP5、主动外联流量趋势、主动外联目的地域分布、主动外联-应用分布、主动外联源IP TOP5和主动外联目的IP TOP5等指标。
- **CFW访问日志中心：**该仪表盘主要展示CFW日志的互联网访问-拦截趋势、主动外联-拦截趋势、互联网阻断应用TOP5、互联网阻断目的TOP5、互联网阻断来源TOP5、主动外联阻断应用TOP5、主动外联阻断目的TOP5和主动外联阻断来源TOP5等指标。
- **CFW攻击日志中心**：该仪表盘主要展示CFW日志的攻击趋势、攻击来源分布、TOP5 执行命令、攻击目的TOP5和攻击来源TOP5等指标。

CSE 仪表盘模板

当日志流标签为log_type=CSE或结构化配置了Microgateway模板时，可使用CSE仪表盘模板查看指标。CSE仪表盘模板分组里有三种仪表盘模板，分别是CSE访问中心、CSE监控中心、CSE秒级监控。更多内容请参考**CSE仪表盘模板**。

- **CSE访问中心：**该仪表盘支持通过上游IP或trace_id过滤信息。主要展示CSE日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、今日PV/UV、Host访问TOP10等指标。
- **CSE监控中心**：该仪表盘支持通过上游IP或trace_id过滤信息。主要展示CSE日志的访问量PV、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP等指标。
- **CSE秒级监控**：该仪表盘支持通过上游IP或trace_id过滤信息。主要展示CSE日志的QPS、成功率、延迟等指标。

DCS 仪表盘模板

当日志流标签为log_type=dcs_audit或结构化配置了DCS_AUDIT模板时，可使用DCS仪表盘模板查看指标。DCS仪表盘模板分组里有DCS审计日志中心仪表盘模板。更多内容请参考**DCS仪表盘模板**。

DCS审计日志中心：访问用户数、访问客户端数、审计日志条数、平均响应时间、平均QPS、TOP5 用户、TOP5 客户端、TOP5 执行命令、热Key、审计日志详情等指标。

DDS 仪表盘模板

当日志流标签为log_type=dds_audit或结构化配置了DDS_AUDIT模板时，可使用DDS仪表盘模板查看指标。DDS仪表盘模板分组里有DDS审计日志中心仪表盘模板。更多内容请参考**DDS仪表盘模板**。

DDS审计日志中心：访问用户数、访问客户端数、审计日志条数、平均响应时间、平均QPS、TOP5 用户、TOP5 客户端、TOP5 执行命令、热Key、审计日志详情等指标。

DMS 仪表盘模板

当日志流标签为log_type=dms_rebalanced或结构化配置了DMS模板时，可使用DMS仪表盘模板查看指标。DMS仪表盘模板分组里有DMS重平衡日志中心。更多内容请参考**DMS仪表盘模板**。

DMS重平衡日志中心：该仪表盘主要展示DMS重平衡消费组个数、重平衡次数、消费组重平衡次数、重平衡原因及组详情等指标。

DSL 仪表盘模板

当日志流标签为log_type=dsl或结构化配置了DSL系统日志模板时，可使用DSL仪表盘模板查看指标。DSL仪表盘模板分组里有DSL加工任务监控中心。更多内容请参考**DSL仪表盘模板**。

DSL加工任务监控中心：该仪表盘主要展示输入行数、输出行数、过滤行数、失败行数等指标。

ELB 仪表盘模板

当日志流标签为log_type=elb_7layer_access或结构化配置了ELB模板时，可使用ELB仪表盘模板查看指标。ELB仪表盘模板分组里有三种仪表盘模板，分别是ELB监控中心、ELB访问中心和ELB秒级监控。更多内容请参考**ELB仪表盘模板**。

- **ELB7层监控中心**：该仪表盘支持通过负载均衡器、客户端IP、后端服务器IP或弹性IP地址过滤信息。主要展示ELB日志的访问量PV、访问量UV、流量、访问失败率、延迟、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP等指标。
- **ELB7层访问中心**：该仪表盘支持通过负载均衡器、客户端IP、后端服务器IP或弹性IP地址过滤信息。主要展示ELB日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、访问量UV分布（中国）、访问量UV分布（世界）、平均时延分布（中国）、平均时延分布（世界）、今日PV/UV、7日PV/UV、区域访问TOP10(省份)、区域访问TOP10(城市)、Host访问TOP10、UserAgent访问TOP10、设备占比(终端)、设备占比(系统)、TOP URL、TOP 访问IP等指标。
- **ELB7层秒级监控**：该仪表盘支持通过负载均衡器、客户端IP、后端服务器IP或弹性IP地址过滤信息。主要展示ELB日志的QPS、成功率、延迟、流量、状态码、Upstream状态码等指标。

ER 仪表盘模板

当日志流标签为log_type=er_flow或结构化配置了ER模板时，可使用ER仪表盘模板查看指标。ER仪表盘模板分组里有ER流量日志中心仪表盘模板。更多内容请参考**ELB仪表盘模板**。

ER流量日志中心：该仪表盘主要展示ER流量日志的实例ID、连接ID、流量方向、源IP、目的IP、协议类型、TOP20包数统计、TOP20流量统计、流日志条数和流日志详情。

METRIC 仪表盘模板

当日志流标签为log_type=metric或结构化配置了METRIC系统日志模板时，可使用METRIC仪表盘模板查看指标，METRIC仪表盘模板里有日志生成指标任务监控中心仪表盘模板。更多内容请参考**METRIC仪表盘模板**。

日志生成指标任务监控中心：主要展示日志生成指标的输入行数、输出行数、满足过滤条件行数、不满足过滤条件行数等指标。

NGINX 仪表盘模板

当日志流标签为log_type=nginx_access或结构化配置了NGINX模板时，可使用NGINX仪表盘模板查看指标。NGINX仪表盘模板分组里有三种仪表盘模板，分别是NGINX监控中心、NGINX访问中心和NGINX秒级监控。更多内容请参考**NGINX仪表盘模板**。

- **NGINX监控中心**：该仪表盘主要展示NGINX日志的访问量PV、访问量UV、流量、访问失败率、延迟、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP等指标。
- **NGINX访问中心**：该仪表盘主要展示NGINX日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、访问量UV分布（中国）、访问量UV分布（世界）、平均时延分布（中国）、平均时延分布（世界）、今日PV/UV、7日PV/UV、区域访问TOP10(省份)、区域访问TOP10(城市)、Host访问TOP10、

UserAgent访问TOP10、设备占比(终端)、设备占比(系统)、TOP URL、TOP 访问IP等指标。

- **NGINX秒级监控**：该仪表盘主要展示NGINX日志的QPS、成功率、延迟、流量、状态码、Upstream状态码等指标。

VPC 仪表盘模板

当日志流标签为log_type=vpc_flow或结构化配置了VPC模板时，可使用VPC仪表盘模板查看指标。VPC仪表盘模板分组里有VPC流日志仪表盘模板。更多内容请参考**VPC仪表盘模板**。

VPC流日志：该仪表盘主要展示VPC流日志的Action总次数、ACCEPT总字节数、ACCEPT总包数、REJECT总字节数、REJECT总包数、源地址的Action次数分布、每分钟Action次数、Action分布、流日志记录状态分布、Action次数的源地址运营商分布、Top5字节数的源地址、Top5字节数的目标地址、各协议的每分钟包数和弹性网卡等指标。

WAF 仪表盘模板

当日志流标签为log_type=waf_access（访问日志中心）、log_type=waf_attack（安全日志中心）或结构化配置了WAF模板时，可使用WAF仪表盘模板查看指标。WAF仪表盘模板分组里有两种仪表盘模板，分别是WAF访问日志中心和WAF安全日志中心。更多内容请参考**WAF仪表盘模板**。

- **WAF访问日志中心**：该仪表盘主要展示WAF访问日志的访问量PV、访问量UV、流入流量、网络in带宽峰值、网络out带宽峰值、流量带宽趋势、PV/UV趋势、访问状态分布、访问来源等指标。
- **WAF安全日志中心**：该仪表盘主要展示WAF安全日志的攻击峰值、被攻击网站、Web攻击拦截、CC攻击拦截、攻击者UV、攻击类型分布、CC攻击、Web攻击等指标。

采集诊断仪表盘模板

当日志流标签为log_type=diagnosis时，可使用采集诊断仪表盘模板查看指标。采集诊断仪表盘模板分组里有ICAgent采集监控、ICAgent整体状态、ICAgent异常监控。更多内容请参考**采集诊断仪表盘模板**。

- **ICAgent采集监控**：该仪表盘主要展示采集文件数、采集机器数/同比昨天、采集文件分布、采集机器数等图表。
- **ICAgent整体状态**：该仪表盘主要展示活跃ICAgent数、CPU趋势、ICAgent整体状态等图表。
- **ICAgent异常监控**：该仪表盘主要展示关键错误数、丢弃超大行、请求LTS失败、文件超过上限问题数等图表。

7.3.3.2 APIG 仪表盘模板

云日志服务支持日志采集向导一站式采集APIG日志，进行多维度分析，并为APIG日志配置结构化和仪表盘，仪表盘包括APIG访问中心、APIG秒级监控、APIG监控中心。

7.3.3.2.1 APIG 监控中心

APIG监控中心仪表盘主要展示APIG日志的访问量PV、访问量UV、流量、访问失败率、延迟、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟

TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP等指标。全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

- 已采集APIG日志，详情请参见**APIG接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

APIG（API Gateway）提供高性能、高可用、高安全的API托管服务，能快速将企业服务能力包装成标准API服务，帮助您轻松构建、管理和部署任意规模的API，并上架API云商店进行售卖。借助API网关，可以简单、快速、低成本、低风险地实现内部系统集成、业务能力开放及业务能力变现。API网关帮助您变现服务能力的同时，降低企业研发投入，让您专注于企业核心业务，提升运营效率。

分析APIG监控情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“APIG日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“APIG仪表盘模板>APIG监控中心”仪表盘，查看图表详情。

----结束

APIG监控中心仪表盘中的过滤器说明如下所示：

- 获取所有请求域名，所关联的查询分析语句如下所示：
```
select distinct(host)
```
- 获取所有app_id，所关联的查询分析语句如下所示：
```
select distinct(app_id)
```

重要图表说明

APIG监控中心仪表盘中重要图表说明如下所示：

- **访问量PV**图展示访问量的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss' ) as _time_,PV FROM ( SELECT TIME_CEIL ( TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'), 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log GROUP BY _time_ )
```
- **请求成功率**图展示请求成功率的变化情况，所关联的查询分析语句如下所示：
```
select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
```
- **平均延迟**图展示平均延迟的变化情况，所关联的查询分析语句如下所示：
```
select round(avg(request_time) * 1000, 3) as cnt
```
- **4XX请求数**图展示4xx的请求状态码的变化情况，所关联的查询分析语句如下所示：
```
SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
```
- **404请求数**图展示404请求状态码的变化情况，所关联的查询分析语句如下所示：
```
SELECT COUNT(1) as cnt WHERE "status" = 404
```
- **499请求数**图展示499请求状态码的变化情况，所关联的查询分析语句如下所示：
```
SELECT COUNT(1) as cnt WHERE "status" = 499
```
- **504请求数**图展示504请求状态码的变化情况，所关联的查询分析语句如下所示：
```
SELECT COUNT(1) as cnt WHERE "status" = 504
```


- **5XX请求数**图展示5xx的请求数的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,cnt FROM ( SELECT TIME_CEIL ( TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'), 'PT300S' ) AS _time_, count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ )
```
- **状态码分布**图展示请求状态码的变化情况，所关联的查询分析语句如下所示：
```
SELECT status, COUNT(1) AS rm GROUP BY status
```
- **访问量UV**图展示访问量UV的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss' ) as _time_,UV FROM (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(my_remote_addr) as UV  from log group by _time_)
```
- **流量**图展示入流量和出流量的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss') AS _time_,round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量",round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_,sum(request_length) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by  _time_)
```
- **访问失败率**图展示访问失败率和5xx比例的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_,sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >=500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by _time_)
```
- **延迟**图展示访问P50、P90、P99、P9999延迟的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') as _time_,avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as "P90" ,APPROX_QUANTILE_DS("request_time", 0.99)*1000 as 'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by  _time_)
```
- **Host请求TOP**图展示主机请求TOP信息的变化情况，所关联的查询分析语句如下所示：
```
SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY "host" ) ORDER BY pv DESC
```
- **Host延迟TOP**图展示主机延迟TOP信息的变化情况，所关联的查询分析语句如下所示：
```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```
- **Host失败率TOP**图展示主机访问失败率TOP的变化情况，所关联的查询分析语句如下所示：
```
SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS
```



```
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY "host"  ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图展示URL请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT upstream_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT upstream_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY upstream_uri  ) ORDER BY pv desc
```

- **URL失败率TOP**图展示URL失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT upstream_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' GROUP BY upstream_uri  ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图展示后端请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( my_remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY addr  having length(my_remote_addr) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图展示后端延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT my_remote_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' and "my_remote_addr" != '-' GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图展示后端失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT my_remote_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' and "my_remote_addr" != '-' GROUP BY addr) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP**图展示URL延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT upstream_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0
```



```
END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT upstream_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY upstream_uri  ) ORDER BY "平均延迟(ms)" desc
```

7.3.3.2.2 APIG 访问中心

APIG访问中心仪表盘主要展示APIG日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、访问量UV分布（中国）、访问量UV分布（世界）、平均时延分布（中国）、平均时延分布（世界）、今日PV/UV、7日PV/UV、区域访问TOP10(省份)、区域访问TOP10(城市)、Host访问TOP10、UserAgent访问TOP10、设备占比(终端)、设备占比(系统)、TOP URL、TOP 访问IP等信息，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

- 已采集APIG日志，详情请参见**APIG接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

APIG（API Gateway）提供高性能、高可用、高安全的API托管服务，能快速将企业服务能力包装成标准API服务，帮助您轻松构建、管理和部署任意规模的API，并上架API云商店进行售卖。借助API网关，可以简单、快速、低成本、低风险地实现内部系统集成、业务能力开放及业务能力变现。API网关帮助您变现服务能力的同时，降低企业研发投入，让您专注于企业核心业务，提升运营效率。

分析APIG监控情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“APIG日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“APIG仪表盘模板>APIG访问中心”仪表盘，查看图表详情。

----结束

APIG访问中心仪表盘中的过滤器说明如下所示：

- 获取所有请求域名，所关联的查询分析语句如下所示：
  ```
  select distinct(host)
  ```
- 获取所有app_id，所关联的查询分析语句如下所示：
  ```
  select distinct(app_id)
  ```

重要图表说明

- **PV对比昨日**图展示PV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
  ```
- **PV对比上周**图展示PV数对比上周的变化情况，所关联的查询分析语句如下所示
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
  ```
- **UV对比昨日**图展示UV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
  ```

- **UV对比上周**图展示UV数对比上周的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
```
- **访问量PV分布(中国)**图展示中国区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
```
select ip_to_province(my_remote_addr) as province, sum(ori_pv) as pv from (select my_remote_addr, count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000)  where IP_TO_COUNTRY (my_remote_addr) = '中国'  group by province HAVING province not in ('','保留地址','*')
```
- **访问量PV分布(世界)**图展示世界区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
```
SELECT ip_to_country(my_remote_addr) as country,sum(ori_pv) as PV from (select my_remote_addr, count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000) GROUP BY country HAVING country not in ('','保留地址','*')
```
- **平均时延分布（中国）**图展示中国区域内平均时延的分布情况，所关联的查询分析语句如下所示：
```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(my_remote_addr) as province,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000) WHERE  IP_TO_COUNTRY (my_remote_addr) = '中国' GROUP BY province )
 where province not in ('','保留地址','*')
```
- **平均时延分布（世界）**图展示世界区域内平均时延的分布情况，所关联的查询分析语句如下所示：
```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(my_remote_addr) as country,sum(rt)/sum(ori_pv)  * 1000 AS "平均延迟(ms)" from (select my_remote_addr, sum(request_time) as rt,count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000) GROUP BY country )
where  country not in ('','保留地址','*')
```
- **今日PV/UV**图展示今日的PV/UV数，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT600S') AS _time_ , count(1) as PV,  APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log WHERE __time <= CURRENT_TIMESTAMP  and __time >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_)
```
- **区域访问TOP10(省份)**图展示访问排名前十的省份，所关联的查询分析语句如下所示：
```
select ip_to_province(my_remote_addr) as "province", sum(ori_pv) as "访问次数" from(select my_remote_addr, count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000)group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```
- **区域访问TOP10(城市)**图展示访问排名前十的城市，所关联的查询分析语句如下所示：
```
select ip_to_city(my_remote_addr) as "city", sum(ori_pv) as "访问次数" from(select my_remote_addr, count(1) as ori_pv
 group by my_remote_addr
 ORDER BY ori_pv desc
 LIMIT 10000) group by "city" HAVING  "city" <> '-1' order by "访问次数" desc  limit 10
```
- **Host访问TOP10**图展示访问排名前十的主机，所关联的查询分析语句如下所示：
```
select  host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```


- **UserAgent访问TOP10**图展示访问排名前十的UserAgent，所关联的查询分析语句如下所示：
```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)**图展示各终端设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by  type
```
- **设备占比(系统)**图展示各系统设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by  type HAVING type != 'other'
```
- **TOP URL**图展示访问前十url的PV、UV及访问成功率等信息，所关联的查询分析语句如下所示：
```
select router_uri , count(1) as pv, APPROX_COUNT_DISTINCT(my_remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end  ) * 100.0 / count(1), 2) as "访问成功率" group by router_uri ORDER by pv desc
```
- **TOP 访问IP**图展示访问前十的IP及城市、运营商和PV等数据，所关联的查询分析语句如下所示：
```
select my_remote_addr as "来源IP",ip_to_country(my_remote_addr) as "国家",ip_to_province(my_remote_addr) as "省份",ip_to_city(my_remote_addr) as "城市",ip_to_provider(my_remote_addr) as "运营商",count(1) as "PV" group by my_remote_addr ORDER by "PV" desc limit 100
```

7.3.3.2.3 APIG 秒级监控

APIG秒级监控仪表盘主要展示Upstream状态码等信息，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

- 已采集APIG日志，详情请参见**APIG接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

APIG（API Gateway）提供高性能、高可用、高安全的API托管服务，能快速将企业服务能力包装成标准API服务，帮助您轻松构建、管理和部署任意规模的API，并上架API云商店进行售卖。借助API网关，可以简单、快速、低成本、低风险地实现内部系统集成、业务能力开放及业务能力变现。API网关帮助您变现服务能力的同时，降低企业研发投入，让您专注于企业核心业务，提升运营效率。

分析APIG监控情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“APIG日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“APIG仪表盘模板>APIG秒级监控”仪表盘，查看图表详情。

----结束

APIG秒级监控仪表盘中的过滤器说明如下所示：

- 获取所有请求域名，所关联的查询分析语句如下所示：
```
select distinct(host)
```


- 获取所有app_id，所关联的查询分析语句如下所示：
  ```
  select distinct(app_id)
  ```

重要图表说明

APIG秒级监控仪表盘中的重要图表说明如下所示：

- **QPS**图展示QPS的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT1S'),'yyyy-MM-dd HH:mm:ss') AS _time_ , COUNT(*) as QPS from log group by _time_
  ```
- **成功率**图展示成功率的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
  ```
- **延迟**图展示访问延时的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time, avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from log group by __time)
  ```
- **流量**图展示请求流量和返回body流量的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round( CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 3 ) AS "请求流量",round( CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 3 ) AS "返回body流量" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time , sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量" group by __time)
  ```
- **状态码**图展示响应状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_CEIL ( TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ), 'PT5S' ) AS "time", SUM( CASE WHEN "status" >= 200 AND "status" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "status" >= 300 AND "status" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "status" >= 400 AND "status" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "status" >= 500 AND "status" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "status" < 200 OR "status" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log  WHERE TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time"  ORDER BY "time" ASC LIMIT 100000
  ```
- **后端响应码**图展示后端响应状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_CEIL ( TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ), 'PT5S' ) AS "time", SUM( CASE WHEN "upstream_status" >= 200 AND "upstream_status" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "upstream_status" >= 300 AND "upstream_status" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "upstream_status" >= 400 AND "upstream_status" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "upstream_status" >= 500 AND "upstream_status" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "upstream_status" < 200 OR "upstream_status" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log  WHERE TIME_PARSE ( time_local, 'dd/MMM/yyyy:HH:mm:ss ZZ' ) IS NOT NULL GROUP BY "time"  ORDER BY "time" ASC LIMIT 100000
  ```

7.3.3.3 CCE 仪表盘模板

日志服务支持采集CCE日志，并进行多维度分析。CCE仪表盘模板支持多种仪表盘模板，分别是CCE日志节点操作、CCE日志K8s事件中心、CCE日志聚合检索、CCE日志K8s对象操作、CCE日志账号操作审计、CCE审计日志中心和CCE日志K8s事件查询。

说明

CCE仪表盘功能目前仅对“华北-北京四”、“华东-上海一”和“华南-广州”局点开放。

7.3.3.3.1 CCE 日志节点操作

CCE日志节点操作仪表盘主要展示节点数趋势、非系统用户操作趋势、create操作状态码分布、delete操作状态码分布等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志节点操作”仪表盘，查看图表详情。

----结束

CCE日志节点操作仪表盘中的过滤器说明如下所示：

- 节点名称，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.name")
  ```
- 操作用户，所关联的查询分析语句如下所示：
  ```
  select distinct("user.username")
  ```
- 状态码，所关联的查询分析语句如下所示：
  ```
  select distinct("responseStatus.code")
  ```
- 操作类型，所关联的查询分析语句如下所示：
  ```
  select distinct("verb")
  ```

重要图表说明

CCE日志节点操作仪表盘中重要图表说明如下所示：

- 节点数趋势，所关联的查询分析语句如下所示：
  ```
  SELECT  time_series( TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd''T''HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ) as "dt",  count(DISTINCT("objectRef.name")) as "节点数" where "objectRef.resource" = 'nodes'  and "objectRef.subresource" = 'status'  and "verb" in ('update', 'patch') and "user.username" = 'system:node' group by  "dt" order by  "dt" desc limit  10000
  ```
- 非系统用户操作趋势，所关联的查询分析语句如下所示：
  ```
  SELECT time_series( TIME_PARSE(LEFT(requestReceivedTimestamp, 23),'yyyy-MM-dd''T''HH:mm:ss.SSS'), 'PT1H', 'yyyy-MM-dd HH', '0' ) as "dt", count(*) as "请求", "user.username" where "objectRef.resource" = 'nodes' and "user.username" not in ( 'kube-controller-manager','kube-apiserver-kubelet-client','apiserver') and "user.username" not like  'system:%' and "verb" in ('create','delete','update','patch') group by "dt", "user.username" order by "dt","请求" desc limit 10000
  ```
- create操作状态码分布，所关联的查询分析语句如下所示：
  ```
  select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'create' group by "状态码"
  ```
- delete操作状态码分布，所关联的查询分析语句如下所示：
  ```
  select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'delete' group by "状态码"
  ```
- patch操作状态码分布，所关联的查询分析语句如下所示：
  ```
  select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'patch' group by "状态码"
  ```
- update操作状态码分布，所关联的查询分析语句如下所示：
  ```
  select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" = 'update' group by "状态码"
  ```


- 节点封锁/解除封锁操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*)  as "count" where "requestObject" in ('{"spec":{"unschedulable":false}}','{"spec":{"unschedulable":true}}')  group by "状态码"
```
- Label操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" in ('patch','update') and "requestObject" = 'labels' and "requestObject" = 'metadata' group by "状态码"
```
- Taint操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.resource" = 'nodes' and "verb" in ('patch','update') and "requestObject" = 'taints' group by "状态码"
```
- 驱逐操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "objectRef.subresource" = 'eviction' and "objectRef.resource" = 'pods' and "verb" = 'create' group by "状态码"
```
- 节点增删操作列表，所关联的查询分析语句如下所示：
```
select  "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "verb" AS "操作动作", "stageTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "objectRef.resource" = 'nodes' and "verb" in ('create','delete')
```
- Taint操作列表，所关联的查询分析语句如下所示：
```
select  "auditID" AS "Audit ID", "objectRef.name" AS "节点名","requestObject" AS "Taints", "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码"  where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'taints'
```
- 驱逐操作列表，所关联的查询分析语句如下所示：
```
select  "auditID" AS "Audit ID", "objectRef.name" AS "pod", "sourceIPs" AS "源地址", "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码"  where "objectRef.resource" = 'pods' and "verb" = 'create' and "objectRef.subresource" = 'eviction'
```
- Label操作列表，所关联的查询分析语句如下所示：
```
select  "auditID" AS "Audit ID", "objectRef.name" AS "节点名", "requestObject" AS "Label", "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码"  where "objectRef.resource" = 'nodes' and "verb" = 'patch' and "requestObject" = 'labels'
```
- 封锁操作列表，所关联的查询分析语句如下所示：
```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名",  "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch' and "objectRef.resource" = 'nodes' and "requestObject" ='true' and "requestObject" = 'unschedulable'
```
- 取消封锁操作列表，所关联的查询分析语句如下所示：
```
select "auditID" AS "Audit ID", "objectRef.name" AS "节点名",  "requestReceivedTimestamp" AS "操作时间", "user.username" AS "操作账号", "responseStatus.code" AS "状态码" where "verb" = 'patch' and "objectRef.resource" = 'nodes' and "requestObject" not in ('true','taints','unschedulable')
```

7.3.3.3.2 CCE 日志 K8s 对象操作

CCE日志K8s对象操作仪表盘主要展示重要操作趋势、非系统用户操作趋势、create操作资源类型分布、delete操作资源类型分布等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志K8s对象操作”仪表盘，查看图表详情。

----结束

CCE日志K8s对象操作仪表盘中的过滤器说明如下所示：

- 命名空间，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.namespace")
  ```
- 操作类型，所关联的查询分析语句如下所示：
  ```
  select distinct("verb")
  ```
- 状态码，所关联的查询分析语句如下所示：
  ```
  select distinct("responseStatus.code")
  ```
- 资源对象，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.name")
  ```
- 资源类型，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.resource")
  ```
- 操作用户，所关联的查询分析语句如下所示：
  ```
  select distinct("user.username")
  ```

重要图表说明

CCE日志K8s对象操作仪表盘中重要图表说明如下所示：

- 重要操作趋势展示创建、升级、更新、删除等重要操作的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT REPLACE(LEFT(requestReceivedTimestamp, 16),'T',' ') AS "dt", "verb" as "操作类型", count(*) as "count" where "verb" in ('create','delete','update','patch') and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "dt","操作类型" order by "dt" limit 10000
  ```
- 非系统用户操作趋势，所关联的查询分析语句如下所示：
  ```
  SELECT REPLACE(LEFT(requestReceivedTimestamp, 16),'T',' ') AS "dt", count(*) as "请求次数","user.username" WHERE "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-client','apiserver') and "user.username" not like 'system:%' and "verb" in ('create','delete','update','patch') and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresss','configmaps','secrets','pvcs') group by "dt", "user.username" limit 10000
  ```
- create操作资源类型分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
  ```
- delete操作资源类型分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
  ```
- update操作资源类型分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
  ```
- patch操作资源类型分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by "objectRef.resource"
  ```


- create操作用户分布，所关联的查询分析语句如下所示：
```
select "user.username" as "操作用户", count(*)  as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "user.username"
```
- delete操作用户分布，所关联的查询分析语句如下所示：
```
select "user.username" as "操作用户", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "user.username"
```
- update操作用户分布，所关联的查询分析语句如下所示：
```
select "user.username" as "操作用户", count(*)  as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "user.username"select "user.username" as "操作用户", count(*)  as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "user.username"
```
- patch操作用户分布，所关联的查询分析语句如下所示：
```
select "user.username" as "操作用户", count(*)  as "count"  where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "user.username"
```
- create操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "responseStatus.code"
```
- delete操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "responseStatus.code"
```
- update操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "responseStatus.code"
```
- patch操作状态码分布，所关联的查询分析语句如下所示：
```
select cast("responseStatus.code" as varchar) as "状态码", count(*)  as "count" where "verb" = 'patch' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims') group by  "responseStatus.code"
```
- create操作趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'create' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims')  group by dt, "objectRef.resource" order by dt limit 10000
```
- delete操作趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'delete' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims')  group by dt, "objectRef.resource" order by dt limit 10000
```
- update操作趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'update' and "objectRef.resource" in ('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','configmaps','persistentvolumeclaims')  group by dt, "objectRef.resource" order by dt limit 10000
```
- patch操作趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" where "verb" = 'patch' and "objectRef.resource" in
```



```
('deployments','statefulsets','cronjobs','daemonsets','jobs','pods','services','ingresses','configmaps','config
maps','persistentvolumeclaims')  group by dt, "objectRef.resource" order by dt limit 10000
```

7.3.3.3.3 CCE 日志 K8s 事件查询

CCE日志K8s事件查询仪表盘主要展示事件总数、普通事件数、重要事件分布、警告事件数等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志K8s事件查询”仪表盘，查看图表详情。

----结束

CCE日志K8s事件查询仪表盘中的过滤器说明如下所示：

- 事件等级分为Warning和Normal。
- 事件类型，所关联的查询分析语句如下所示：
  ```
  select distinct("name")
  ```
- 集群ID，所关联的查询分析语句如下所示：
  ```
  select distinct("cluster_id")
  ```
- 命名空间，所关联的查询分析语句如下所示：
  ```
  select distinct("namespace")
  ```
- 名称，所关联的查询分析语句如下所示：
  ```
  select distinct("resource_name")
  ```

重要图表说明

CCE**日志K8s事件查询**仪表盘中重要图表说明如下所示：

- 事件总数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', '' ) as "_time_","total" FROM ( SELECT
  TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log GROUP BY "_time_" )
  ```
- 普通事件数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', '' ) as "_time_","total" FROM ( SELECT
  TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log where "type" = 'Normal'
  GROUP BY "_time_" )
  ```
- 警告事件数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', '' ) as "_time_","total" FROM ( SELECT
  TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log where "type" = 'Warning'
  GROUP BY "_time_" )
  ```
- 错误事件数，所关联的查询分析语句如下所示：


```
SELECT TIME_FORMAT( "_time_", 'yyyy-MM-dd HH:mm:ss', '' ) as "_time_","total" FROM ( SELECT TIME_CEIL ( __time, 'PT1H' ) AS "_time_", count( 1 ) AS "total" FROM log  where "type" = 'Error' GROUP BY "_time_" )
```

- 重要事件分布，所关联的查询分析语句如下所示：
```
select "name",count(1) as 'total' from log group by "name"
```
- 重要事件统计，所关联的查询分析语句如下所示：
```
SELECT "type" as "等级", "name" as "类型", count(1) as "总数", "reason" as "详细内容" from log where "type" != 'Normal'  group by "type", "name", "reason" order by "总数" desc
```
- Pod重要事件统计，所关联的查询分析语句如下所示：
```
SELECT "type" as "等级", "name" as "类型",  count(1) as "总数", "reason" as "详细内容"  from log where "type" != 'Normal'  and "resource_kind" = 'Pod' group by  "type",  "name", "reason" order by "总数" desc
```
- 最近100条事件，所关联的查询分析语句如下所示：
```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as "Time","type" as "等级", "name" as "类型", "reason" as "详细
```

7.3.3.3.4 CCE 日志 K8s 事件中心

CCE日志K8s事件中心仪表盘主要展示节点FD不足、节点磁盘空间不足、事件同步异常、事件分布等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志K8s事件中心”仪表盘，查看图表详情。

----结束

CCE日志K8s事件中心仪表盘中的过滤器说明如下所示：

- 事件等级分为Warning和Normal。
- 事件类型，所关联的查询分析语句如下所示：
```
select distinct("name")
```
- 集群ID，所关联的查询分析语句如下所示：
```
select distinct("cluster_id")
```
- 命名空间，所关联的查询分析语句如下所示：
```
select distinct("namespace")
```
- 名称，所关联的查询分析语句如下所示：
```
select distinct("resource_name")
```

重要图表说明

CCE日志K8s事件中心仪表盘中重要图表说明如下所示：

- Conntrack Full，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'ConntrackFull'  ) )
```
- 事件同步异常，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'NTPIsDown') )
```
- 节点Pid不足，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name" in ('PIDPressure','NodeHasPIDPressure') ) )
```
- 节点FD不足，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'NodeHasFDPressure') )
```
- 节点磁盘空间不足，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'NodeHasDiskPressure') )
```
- Pod OOM，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where   "reason" in ('OOMKilling','PodOOMKilling')) )
```
- DockerHung，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'Failed' and "reason" = 'DockerHung') )
```
- 节点重启，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'NodeRebooted') )
```
- 镜像拉取失败，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'Failed' and "reason" = 'ImagePullBackOff') )
```
- 节点OOM，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name" = 'SystemOOM') )
```
- Pod启动失败，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( "total", 3600) as diff from( select count(1) as "total" from log where  "name"= 'Failed' and "resource_kind" = 'Pod' and  "reason" = 'ImagePullBackOff') )
```
- 事件分布，所关联的查询分析语句如下所示：
```
select "type", count(*) as "事件数" group by "type"
```
- Warning事件趋势，所关联的查询分析语句如下所示：
```
select time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt",count(1) as "count"  from log  where "type" = 'Warning'  group by "dt" order by "dt"
```
- Error事件趋势，所关联的查询分析语句如下所示：
```
select time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "dt",count(1) as "count" from log  where "type" = 'Error' group by "dt" order by "dt"
```
- Pod OOM事件列表，所关联的查询分析语句如下所示：
```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "Time", "resource_kind" as "事件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where "name" in ('OOMKilling','PodOOMKilling') order by __time desc limit 100
```
- Pod驱动事件列表，所关联的查询分析语句如下所示：


```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as "Time", "resource_kind" as "事件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where "name" = 'NodeControllerEviction' order by __time desc limit 100
```

- 重要事件列表，所关联的查询分析语句如下所示：

```
select TIME_FORMAT( __time, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as "Time", "type" as "等级", "resource_kind" as "事件目标", "name" as "类型", "resource_name" as "目标名", "reason" as "详细内容" from log where "type" in ('Warning','Error') order by __time desc limit 100
```

7.3.3.3.5 CCE 日志聚合检索

CCE日志聚合检索仪表盘主要展示用户分布趋势、命名空间分布趋势、操作类型分布趋势、状态码分布趋势等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志聚合检索”仪表盘，查看图表详情。

----结束

CCE日志聚合检索仪表盘中的过滤器说明如下所示：

- 命名空间，所关联的查询分析语句如下所示：

```
select distinct("objectRef.namespace")
```

- 操作用户，所关联的查询分析语句如下所示：

```
select distinct("user.username")
```

- 状态码，所关联的查询分析语句如下所示：

```
select distinct("responseStatus.code")
```

- 操作类型，所关联的查询分析语句如下所示：

```
select distinct("verb")
```

- 资源对象，所关联的查询分析语句如下所示：

```
select distinct("objectRef.name")
```

- 资源类型，所关联的查询分析语句如下所示：

```
select distinct("objectRef.resource")
```

- 请求URL，所关联的查询分析语句如下所示：

```
select distinct("requestURI")
```

- userAgent，所关联的查询分析语句如下所示：

```
select distinct("userAgent")
```

重要图表说明

CCE**日志聚合检索**仪表盘中重要图表说明如下所示：

- 操作用户分布趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "user.username" as "操作用户", count(*) as "count" group by dt, "user.username" order by dt limit 10000
```
- 命名空间分布趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.namespace" as "命名空间", count(*) as "count"  group by dt, "objectRef.namespace" order by dt limit 10000
```
- 操作类型分布趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.namespace" as "命名空间", count(*) as "count"  group by dt, "objectRef.namespace" order by dt limit 10000
```
- 状态码分布趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, cast("responseStatus.code" as varchar) as "返回码", count(*)  as "count" group by dt, "返回码" order by dt limit 10000
```
- 资源类型分布趋势，所关联的查询分析语句如下所示：
```
SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" group by dt, "objectRef.resource" order by dt limit 10000    SELECT REPLACE(LEFT(stageTimestamp, 16),'T',' ') AS dt, "objectRef.resource" as "资源类型", count(*) as "count" group by dt, "objectRef.resource" order by dt limit 10000
```
- 重要操作列表，所关联的查询分析语句如下所示：
```
select  "auditID" AS "Audit ID", "verb" AS "操作类型", "requestReceivedTimestamp" AS "开始时间", "stageTimestamp" AS "结束时间", "user.username" AS "操作账号", "sourceIPs" AS "操作源","userAgent","objectRef.namespace" AS "命名空间", CONCAT(CONCAT("objectRef.resource", '/'), "objectRef.subresource") AS "操作对象", "objectRef.name" AS "资源名", "responseStatus.code" AS "返回码"
```

7.3.3.3.6 CCE 日志账号操作审计

CCE日志账号操作审计仪表盘主要展示资源创建数、资源修改数、资源删除数、操作命名空间分布等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志账号操作审计”仪表盘，查看图表详情。

----结束

CCE日志账号操作审计仪表盘中的过滤器说明如下所示：

- 用户名，所关联的查询分析语句如下所示：
```
select distinct("user.username")
```
- 命名空间，所关联的查询分析语句如下所示：
```
select distinct("objectRef.namespace")
```
- 状态码，所关联的查询分析语句如下所示：
```
select distinct("responseStatus.code")
```


重要图表说明

CCE日志账号操作审计仪表盘中重要图表说明如下所示：

- 资源创建数，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(1) as total from log where "verb" = 'create') )
```
- 资源修改数，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(*)  as "total" from log where "verb" in ('update','patch')) )
```
- 资源删除数，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(*)  as "total" from log where "verb" = 'delete') )
```
- 操作命名空间分布，所关联的查询分析语句如下所示：
```
select case when "objectRef.namespace" is null then '_all__' else "objectRef.namespace" end as ns, count(1) as total group by  ns  limit 10000
```
- 删除资源分布，所关联的查询分析语句如下所示：
```
SELECT "objectRef.resource" as "resource", count(1) as "count" where  "verb" = 'delete' group by "resource"
```
- 操作轨迹，所关联的查询分析语句如下所示：
```
select case  when "操作" is null then '无' else "操作" end as "操作", "时间", v from  (select concat(CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else "objectRef.subresource"  end, '[', verb, ']'  ) as "操作", time_series(__time, 'PT1H', 'yyyy-MM-dd HH', '0') as "时间", count(1) as v  from  log where "verb" in ('create', 'patch',  'update', 'delete')  group by "操作", "时间" order by "时间" desc  limit  10000  )
```
- 资源操作分布，所关联的查询分析语句如下所示：
```
select CASE WHEN "objectRef.subresource" is null then "objectRef.resource" else "objectRef.subresource" end as "资源", verb as "操作", count(1) as total where "verb" in ('create','update','patch','delete') group by "资源", "操作" limit 10000
```
- 创建资源列表，所关联的查询分析语句如下所示：
```
SELECT "auditID" as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间", "requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码","sourceIPs" as "源地址", "requestObject" as "详细内容" where "verb" = 'create' order by __time desc limit 1000
```
- 修改资源列表，所关联的查询分析语句如下所示：
```
SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间","requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码","sourceIPs" as "源地址", requestObject as "详细内容" where "verb" in ('upate','patch') order by __time desc limit 1000
```
- 资源访问列表，所关联的查询分析语句如下所示：
```
SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间", "requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码","sourceIPs" as "源地址", requestObject as "详细内容" where "verb" in ('get','list') order by __time desc limit 1000
```
- 资源删除列表，所关联的查询分析语句如下所示：
```
SELECT auditID as "事件ID", time_format("__time",'yyyy-MM-dd HH:mm:ss') as "操作时间", "requestURI" as "资源", "objectRef.name" as "资源名", "responseStatus.code" as "状态码","sourceIPs" as "源地址", requestObject as "详细内容" where "verb" = 'delete' order by __time desc limit 1000
```

7.3.3.3.7 CCE 日志审计中心

CCE日志审计中心仪表盘主要展示总审计记录数、操作用户数、活跃节点数、异常访问次数等。

前提条件

- 已采集CCE日志，详情请参见**云容器引擎CCE应用日志接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云容器引擎（Cloud Container Engine，简称CCE）提供高度可扩展的、高性能的企业级Kubernetes集群。借助云容器引擎，您可以在华为云上轻松部署、管理和扩展容器化应用程序。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CCE日志审计中心”仪表盘，查看图表详情。

----结束

CCE日志审计中心仪表盘中的过滤器说明如下所示：

- 命名空间，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.namespace")
  ```
- 操作用户，所关联的查询分析语句如下所示：
  ```
  select distinct("user.username")
  ```
- 操作类型，所关联的查询分析语句如下所示：
  ```
  select distinct("verb")
  ```
- 状态码，所关联的查询分析语句如下所示：
  ```
  select distinct("responseStatus.code")
  ```
- 资源对象，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.name")
  ```
- 资源类型，所关联的查询分析语句如下所示：
  ```
  select distinct("objectRef.resource")
  ```
- 请求URL，所关联的查询分析语句如下所示：
  ```
  select distinct("requestURI")
  ```
- UserAgent，所关联的查询分析语句如下所示：
  ```
  select distinct("userAgent")
  ```

重要图表说明

CCE日志审计中心仪表盘中重要图表说明如下所示：

- 总审计记录数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total" , round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(1) as total from log ) )
  ```
- 操作用户数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total" , round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc"  from (select compare( total , 86400) as diff from( select count(distinct("user.username")) as total from log ) )
  ```
- 活跃节点数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(DISTINCT "user.username") as total  from log where "objectRef.resource" = 'nodes' and "objectRef.subresource" = 'status' and "verb" in ('update','put','patch') and "user.username" in ('node','system')) )
  ```
- 异常访问次数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(1) as total from log where "responseStatus.code" >= 400) )
  ```
- 敏感操作次数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total" , round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select compare( total , 86400) as diff from( select count(1)  as "total" from log where ("verb" = 'create' AND "objectRef.subresource" = 'exec') OR ("verb" = 'create' AND "objectRef.subresource" = 'attach'
  ```



```
AND "objectRef.resource" = 'pods') OR ("objectRef.resource" = 'secrets' AND "verb"= 'get' AND
( "user.username" != 'apiserver') AND ("user.username" not like 'system:node:%')) OR ("verb"=
'delete' AND ( "user.username" not like 'system:node:%') AND ( "user.username" not like
'system:serviceaccount:kube-system:%') AND ( "user.username" != 'system:apiserve') AND
( "user.username" != 'system:apiserve') AND ( "user.username" != 'system:kube-scheduler') AND
("user.username" != 'system:kube-controller-manager'))) )
```

- 创建操作次数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select
  compare( total , 86400) as diff from( select count(1) as total from log where verb = 'create') )
  ```
- 更新操作次数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total" , round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc"  from (select
  compare( total , 86400) as diff from( select count(1) as total from log where verb in
  ('update','patch')) )
  ```
- 删除操作次数，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100 , 2 ) as "inc" from (select
  compare( total , 86400) as diff from( select count(1) as total from log where verb = 'delete') )
  ```
- 操作用户分布，所关联的查询分析语句如下所示：
  ```
  select "user.username" as "用户名", count(*) as "count"  group by "用户名" order by "count" desc
  ```
- 命名空间分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.namespace" as "命名空间", count(*) as "count"  group by "命名空间"
  ```
- 资源类型分布，所关联的查询分析语句如下所示：
  ```
  select "objectRef.resource" as "资源类型", count(*) as "count"  group by "资源类型" order by "count"
  desc limit 20
  ```
- 操作类型分布，所关联的查询分析语句如下所示：
  ```
  select verb as "操作类型", count(*) as "count" group by "操作类型" order by "count" desc
  ```
- 节点操作分布，所关联的查询分析语句如下所示：
  ```
  select "verb" as "操作类型", count(*) as "count" where  "objectRef.resource" = 'nodes' AND ("verb" in
  ('create','delete') ) group by "操作类型" order by "count" desc
  ```
- 工作负载操作分布，所关联的查询分析语句如下所示：
  ```
  select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and
  "objectRef.resource" in ('deployments','statefulsets','daemonsets','jobs','cronjobs') group by "操作类型"
  order by "count" desc
  ```
- Service/Ingress操作分布，所关联的查询分析语句如下所示：
  ```
  select "verb" as "操作类型", count(*) as "count" where "verb" in ('create', 'delete') and
  "objectRef.resource" in ('ingressess','services') group by "verb"  order by "count" desc
  ```
- 重要操作趋势，所关联的查询分析语句如下所示：
  ```
  SELECT REPLACE(LEFT("stageTimestamp", 16),'T',' ') AS "dt", "verb", count(*) as "count" where "verb"
  in ('create','delete','update','patch') group by "dt", "verb" order by "dt" limit 10000
  ```
- 非系统用户操作趋势，所关联的查询分析语句如下所示：
  ```
  SELECT REPLACE(LEFT("stageTimestamp", 16),'T',' ') AS "dt", count(*) as "count", "user.username" as
  "用户名称" where "user.username" not in ('kube-controller-manager','kube-apiserver-kubelet-
  client','system','apiserver') group by "dt", "用户名称" order by "dt" limit 10000
  ```

7.3.3.4 CDN 仪表盘模板

云日志服务支持日志采集向导一站式采集CDN日志，进行多维度分析，并为CDN日志配置结构化和仪表盘，仪表盘包括CDN热门资源、CDN用户分析、CDN错误分析。

7.3.3.4.1 CDN 错误分析

CDN错误分析仪表盘主要展示错误域名访问Top5、 错误URI访问Top5、错误请求状态分布、 错误按运营商统计、 错误按客户端统计、 错误按省份统计、 4XX错误详情、5XX错误详情、错误按国家统计。

前提条件

- 已采集CDN日志，详情请参见**CDN接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细日志，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“CDN仪表盘模板>CDN错误分析”仪表盘，查看图表详情。

----结束

重要图表说明

CDN错误分析仪表盘中的重要图表说明如下所示：

- 错误域名访问Top5图展示错误域名访问Top5的变化情况，所关联的查询分析语句如下所示：
```
select domain , count(*) as c where http_code > 400 group by domain order by c desc limit 5
```
- 错误URI访问Top5图展示错误URI访问Top5的变化情况，所关联的查询分析语句如下所示：
```
select uri , count(*) as c where http_code > 400 group by uri order by c desc limit 5
```
- 错误请求状态分布图展示错误请求状态分布的变化情况，所关联的查询分析语句如下所示：
```
select http_code , count(*) as c where http_code > 400 group by http_code order by c desc
```
- 错误按运营商统计图展示错误按运营商统计的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_provider(client_ip) as isp , count(*) as c where http_code > 400 group by isp having ip_to_provider(client_ip) != '' order by c desc limit 10
```
- 错误按客户端统计图展示错误按客户端统计的变化情况，所关联的查询分析语句如下所示：
```
select user_agent as "客户端版本", count(*) as "错误次数" where http_code > 400 group by user_agent order by "错误次数" desc limit 10
```
- 错误按省份统计图展示错误按省份统计的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_province(client_ip) as province , count(*) as c where http_code > 400 and IP_TO_COUNTRY (client_ip) = '中国' group by province order by c desc limit 50
```
- 4XX错误详情图展示4XX错误详情的变化情况，所关联的查询分析语句如下所示：
```
SELECT
    province AS "省份",
    isp AS "运营商",
    c AS "错误次数",
    round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)"
   FROM
    (
    SELECT
     ip_to_province ( client_ip ) AS province,
     ip_to_provider ( client_ip ) AS isp,
     count(*) AS c
    FROM
     log
    WHERE
```



```
    http_code >= 400
    AND http_code < 500
   GROUP BY
    province,
    isp
   HAVING
    (
    ip_to_provider ( client_ip )) != ''
   ORDER BY
    c DESC
   )
  GROUP BY
   province,
   isp,
   c
```

- 5XX错误详情图展示5XX错误详情的变化情况，所关联的查询分析语句如下所示：

```
SELECT
    province AS "省份",
    isp AS "运营商",
    c AS "错误次数",
    round( c * 100.0 / sum( c ), 2 ) AS "错误比率(%)"
   FROM
    (
    SELECT
     ip_to_province ( client_ip ) AS province,
     ip_to_provider ( client_ip ) AS isp,
     count(*) AS c
    FROM
     log
    WHERE
        http_code >= 500
    GROUP BY
     province,
     isp
    HAVING
     (
     ip_to_provider ( client_ip )) != ''
    ORDER BY
     c DESC
    )
   GROUP BY
    province,
    isp,
    c
```

- 错误按国家统计图展示错误按国家统计的变化情况，所关联的查询分析语句如下所示：

```
select ip_to_country(client_ip) as country , count(*) as c where http_code > 400  group by country order by c desc limit 50
```

7.3.3.4.2 CDN 基础数据

CDN基础数据仪表盘主要展示缓存命中率、下载速度、访问状态、访问延时分布、请求带宽、访问次数/人数、访问平均延时、请求命中率。

前提条件

- 已采集CDN日志，详情请参见**CDN接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细日志，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“CDN仪表盘模板>CDN基础数据”仪表盘，查看图表详情。

----结束

重要图表说明

CDN基础数据仪表盘中的重要图表说明如下所示：

- **缓存命中率**图展示缓存命中率统计的变化情况，所关联的查询分析语句如下所示：
```
select round(diff[1],2) as Hit_ratio, round(diff[2],2) as diff, round((diff[3]-1)*100, 2) from (select compare(Hit_ratio, 86400) as diff from (select sum(s) * 100.0/count(*) as Hit_ratio from (select case when hit_info = 'HIT' then 1 else 0 end as s from log)))
```
- **下载速度**图展示下载速度统计的变化情况，所关联的查询分析语句如下所示：
```
select round(diff[1],2) as speed, round(diff[2],2) as diff, round((diff[3]-1)*100, 2) from (select compare(speed, 86400) as diff from (select sum(response_size) * 1.0 /sum(response_time) as speed from log ))
```
- **访问状态**图展示访问状态统计的变化情况，所关联的查询分析语句如下所示：
```
select http_code , count(*) as c group by http_code order by c desc
```
- **访问延时分布**图展示访问延时分布统计的变化情况，所关联的查询分析语句如下所示：
```
select
    case when response_time < 100 then '~100ms'
    when response_time < 500 then '100~500ms'
    when response_time < 1000 then '500ms~1s'
    when response_time < 5000 then '1~5s'
    when response_time < 6000 then '5~6s'
    when response_time < 7000 then '6~7s'
    when response_time < 8000 then '7~8s'
    when response_time < 10000 then '8~10s'
    when response_time < 15000 then '10~15s'
    else '15s~' end as latency ,
    count(*) as cnt
    group by latency
    order by  cnt
```
- **请求带宽**图展示请求带宽统计的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
    sum(response_size) * 8/1000000000.0 as "带宽Gbit/min"
    group by TIME_FLOOR(__time,'PT1M')
    order by TIME_FLOOR(__time,'PT1M')
```
- **访问次数/人数**图展示访问次数/人数统计的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
    count(*) as pv, APPROX_COUNT_DISTINCT(client_ip) as uv group by TIME_FLOOR(__time,'PT1M')
order by TIME_FLOOR(__time,'PT1M')
```
- **访问平均延时**图展示访问平均延时统计的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT (TIME_FLOOR(__time,'PT1M'), 'HH:mm', '+08:00') as thisdate,
    avg(response_time) as "平均延时(ms)" group by TIME_FLOOR(__time,'PT1M') order by
TIME_FLOOR(__time,'PT1M')
```
- **请求命中率**图展示请求命中率变化情况，所关联的查询分析语句如下所示：
```
select
    TIME_FORMAT (TIME_FLOOR(m_time,'PT1M'), 'HH:mm', '+08:00' ) as thisdate ,
    sum(is_hit)*100.0/count(*) as hit_ratio
    from  (select  TIME_FLOOR(__time,'PT1M') as m_time ,  case when hit_info = 'HIT'
     then 1 else 0 end as is_hit from log ) group by m_time order by m_time
```


7.3.3.4.3 CDN 用户分析

CDN用户分析仪表盘主要展示访问次数、访问人数、访问客户端统计、运营商次数统计、访问地区分布、有效访问用户TOP、下载量TOP用户。

前提条件

- 已采集CDN日志，详情请参见**CDN接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细日志，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“CDN仪表盘模板>CDN用户分析”仪表盘，查看图表详情。

----结束

重要图表说明

CDN用户分析仪表盘中的重要图表说明如下所示：

- **访问次数**图展示访问次数百分比，所关联的查询分析语句如下所示：
```
select diff[1] as pv,diff[2] as diff, round(100*(diff[3]-1), 2) from (select compare(pv, 86400) as diff from (select count(*) as pv from log))
```
- **访问人数**图展示访问人数百分比，所关联的查询分析语句如下所示：
```
select diff[1] as uv, diff[2] as diff, round((diff[3]-1)*100, 2) from (select compare(uv , 86400) as diff from (select APPROX_COUNT_DISTINCT(client_ip) as uv from log))
```
- **访问客户端统计**图展示不同访问客户端统计数量占比，所关联的查询分析语句如下所示：
```
select ua as "客户端" , sum(c) as "访问次数" from (select case when strpos(ua , 'iphone') > 1 then 'iphone' when strpos(ua, 'ipad') > 1 then 'ipad' when strpos(ua, 'android') > 1 then 'android' when strpos(ua, 'windows') > 1 then 'windows' when strpos(ua , 'mac') > 1 then 'mac' when strpos(ua, 'linux') > 1 then 'linux' else ua end as ua , c from (select count(*) as c , lower(user_agent) as ua from log group by ua order by c desc limit 2000) ) group by "客户端" order by "访问次数" desc limit 100
```
- **运营商次数统计**图展示不同运营商次数统计数量占比，所关联的查询分析语句如下所示：
```
select ip_to_provider(client_ip) as isp ,count(*) as "访问次数" group by isp order by "访问次数" desc limit 100
```
- **访问地区分布**图展示访问地区分布数量，所关联的查询分析语句如下所示：
```
select ip_to_province(client_ip) as province , count(*) as cnt where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province not in ('','保留地址','*') order by cnt desc limit 100
```
- **有效访问用户TOP**图展示有效访问用户TOP的信息，所关联的查询分析语句如下所示：
```
SELECT CASE WHEN ip_to_country(client_ip) = '上海' THEN concat(client_ip, ' ( shanghai )') WHEN ip_to_province(client_ip) = '' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip) = '内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat( client_ip, ' ( ', ip_to_country(client_ip), '/', ip_to_province(client_ip), '/', CASE WHEN ip_to_city(client_ip) = '-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END, ' ', ip_to_provider(client_ip), ' )' ) END AS client, pv as "总访问数", (pv - success_count) as "错误访问数", round( CASE WHEN "throughput" > 0 THEN "throughput" ELSE 0 END, 1 ) AS "下载总量(GB)"from ( select client_ip, count(*) as pv, sum(response_size) / 1024.0 / 1024 / 1024.0 AS throughput, sum( CASE WHEN http_code < 400 THEN
```



```
1 ELSE 0 END ) AS success_count from log group by client_ip order by success_count desc limit 100 )
```

- **下载量TOP用户**图展示下载量TOP用户的信息，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN ip_to_country(client_ip)='上海' THEN concat(client_ip, ' ( shanghai )') WHEN ip_to_province(client_ip)='' THEN concat(client_ip, ' ( Unknown IP )') WHEN ip_to_provider(client_ip)='内网IP' THEN concat(client_ip, ' ( Private IP )') ELSE concat(client_ip, ' ( ', ip_to_country(client_ip), '/', ip_to_province(client_ip), '/', CASE WHEN ip_to_city(client_ip)='-1' THEN 'Unknown city' ELSE ip_to_city(client_ip) END, ' ',ip_to_provider(client_ip), ' )') END AS client, pv as "总访问数", error_count as "错误访问数" , round( CASE WHEN "throughput" > 0 THEN "throughput" ELSE 0 END, 1 ) AS "下载总量(GB)" from ( select client_ip , count(*) as pv, sum(response_size)/1024.0/1024/1024.0 AS throughput , sum(CASE WHEN http_code > 400 THEN 1 ELSE 0 END) AS error_count from log group by client_ip order by throughput desc limit 100)
```

7.3.3.4.4 CDN 热门资源

CDN热门资源仪表盘主要展示域名访问次数Top5、域名下载流量Top5、热门访问（URI）、热门访问（来源）、全国访问次数分布统计、全国下载网速统计、省份统计、运营商流量和速度、运营商统计。

前提条件

- 已采集CDN日志，详情请参见**CDN接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

CDN（Content Delivery Network，内容分发网络）记录了所有域名（包括已删除域名，如果您开通了企业项目，则已删除域名不支持此功能）被网络用户访问的详细日志，您可以将日志接入LTS，对您的业务资源被访问情况进行详细分析。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“CDN仪表盘模板>CDN热门资源”仪表盘，查看图表详情。

----结束

重要图表说明

CDN热门资源仪表盘中的重要图表说明如下所示：

- **域名访问次数Top5**图展示域名访问次数Top5的变化情况，所关联的查询分析语句如下所示：

```
select domain ,count(*) as cnt group by domain order by cnt desc limit 5
```

- **域名下载流量Top5**图展示域名下载流量Top5的变化情况，所关联的查询分析语句如下所示：

```
select domain , sum(response_size) as "下载总量" group by domain order by "下载总量" desc limit 5
```

- **热门访问（URI）**图展示热门访问URI的变化情况，所关联的查询分析语句如下所示：

```
select uri as URI, "访问次数", "访问人数", round( CASE WHEN "下载总量(GB)" > 0 THEN "下载总量(GB)" ELSE 0 END, 2 ) AS "下载总量(GB)" from ( select uri ,count(*) as "访问次数" , APPROX_COUNT_DISTINCT(client_ip) as "访问人数", sum(response_size)/1024.0/1024.0/1024.0 as "下载总量(GB)" where http_code < 400 group by uri order by "访问次数" desc limit 100)
```

- **热门访问（来源）**图展示热门访问来源的变化情况，所关联的查询分析语句如下所示：


```
select refer_domain as "来源",c as "次数",uv as "人数", round(c * 100.0 / sum(c), 2) as  "百分比%" from (select refer_domain as refer_domain,count(*) as c,APPROX_COUNT_DISTINCT(client_ip) as uv from log where refer_domain != '' group by refer_domain order by c desc limit 100 ) GROUP BY refer_domain, c, uv
```

- **全国访问次数分布统计**图展示全国访问次数分布统计的变化情况，所关联的查询分析语句如下所示：

```
select ip_to_province(client_ip) as province , count(*) as cnt where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province not in ('','保留地址','*') order by cnt desc limit 1000
```

- **全国下载网速统计**图展示全国下载网速统计的变化情况，所关联的查询分析语句如下所示：

```
select province, round( CASE WHEN "speed" > 0 THEN "speed" ELSE 0 END, 3 ) AS "speed" from (select ip_to_province(client_ip) as province , sum(response_size)* 1.0 /(sum(response_time)+1) as "speed" , count(*) as c where IP_TO_COUNTRY (client_ip) = '中国' group by province HAVING province not in ('','保留地址','*') order by c desc limit 40)
```

- **省份统计**图展示省份统计的变化情况，所关联的查询分析语句如下所示：

```
select ip_to_province(client_ip) as "省份" ,count(*) as "访问次数", sum(response_size)/1024.0/1024.0/1024.0 as "下载流量(GB)" , sum(response_size) * 1.0 /sum(response_time) as "下载速度(KB/s)" group by "省份" having ip_to_province(client_ip) != '' order by "下载流量(GB)" desc limit 200
```

- **运营商流量和速度**图展示运营商流量和速度的变化情况，所关联的查询分析语句如下所示：

```
select ip_to_provider(client_ip) as isp , sum(response_size)* 1.0 /(sum(response_time)+1) as "下载速度(KB/s)" , sum(response_size)/1024.0/1024.0/1024.0 as "下载总量(GB)", count(*) as c group by isp having ip_to_provider(client_ip) != '' order by c desc limit 10
```

- **运营商统计**图展示运营商统计的变化情况，所关联的查询分析语句如下所示：

```
select "运营商", "访问次数", round( CASE WHEN "下载流量(GB)" > 0 THEN "下载流量(GB)" ELSE 0 END, 2 ) AS "下载流量(GB)",       round( CASE WHEN "下载速度(KB/s)" > 0 THEN "下载速度(KB/s)" ELSE 0 END, 2 ) AS "下载速度(KB/s)" from (      select ip_to_provider(client_ip) as "运营商" ,count(*) as "访问次数", sum(response_size)/1024.0/1024.0/1024.0 as "下载流量(GB)" , sum(response_size) * 1.0 /sum(response_time) as "下载速度(KB/s)" group by "运营商" having ip_to_provider(client_ip) != '' and "运营商" not in ('*') order by "下载流量(GB)" desc limit 200)
```

7.3.3.5 CFW 仪表盘模板

云日志服务联合华为云防火墙（Cloud Firewall，CFW）提供开箱即用仪表盘模板，帮助您快速高效地进行实时决策分析、设备运维管理以及业务趋势分析。仪表盘包括CFW访问日志中心、CFW流量日志中心、CFW攻击日志中心。

7.3.3.5.1 CFW 访问日志中心

CFW访问日志中心仪表盘主要展示互联网访问-拦截趋势、主动外联-拦截趋势、互联网阻断应用TOP5、互联网阻断目的TOP5、互联网阻断来源TOP5、主动外联阻断应用TOP5、主动外联阻断目的TOP5、主动外联阻断来源TOP5。

前提条件

- 已采集CFW日志，详情请参见**CFW接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云防火墙（Cloud Firewall，CFW）是新一代的云原生防火墙，提供云上互联网边界和VPC边界的防护，包括实时入侵检测与防御、全局统一访问控制、全流量分析可视化、日志审计与溯源分析等，同时支持按需弹性扩容、AI提升智能防御能力、灵活扩展满足云上业务的变化和扩张需求，极简应用让用户快速灵活应对威胁。云防火墙可以通过攻击事件日志查看检测到的危险流量的危险等级、受影响的端口、命中的规则、攻击事件类型等信息；通过访问控制日志查看根据访问控制策略放行或阻断的所有流量，以便更好的调整访问控制策略。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“CFW仪表盘模板>CFW访问日志中心”仪表盘，查看图表详情。

----结束

重要图表说明

CFW访问日志中心仪表盘中的重要图表说明如下所示：

- **互联网访问-拦截趋势**图展示互联网访问-拦截趋势的变化情况，所关联的查询分析语句如下所示：
  ```
  select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time,COUNT(*) as frequency WHERE action='deny' AND direction='out2in' group by t_time order by t_time
  ```
- **主动外联-拦截趋势**图展示主动外联-拦截趋势的变化情况，所关联的查询分析语句如下所示：
  ```
  select time_series(MILLIS_TO_TIMESTAMP(hit_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time,COUNT(*) as frequency WHERE action='deny' AND direction='in2out' group by t_time order by t_time
  ```
- **互联网阻断应用TOP5**图展示互联网阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY app ORDER BY frequency DESC LIMIT 5
  ```
- **互联网阻断目的TOP5**图展示互联网阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
  ```
- **互联网阻断来源TOP5**图展示互联网阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='out2in' GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
  ```
- **主动外联阻断应用TOP5**图展示主动外联阻断应用TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT app, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY app ORDER BY frequency DESC LIMIT 5
  ```
- **主动外联阻断目的TOP5**图展示主动外联阻断目的TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT dst_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY dst_ip ORDER BY frequency DESC LIMIT 5
  ```
- **主动外联阻断来源TOP5**图展示主动外联阻断来源TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT src_ip, COUNT(*) as frequency WHERE action='deny' AND direction='in2out' GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
  ```

7.3.3.5.2 CFW 流量日志中心

CFW流量日志中心仪表盘主要展示互联网访问流量趋势、互联网访问流入地域分布(中国)、互联网访问流入地域分布(世界)、互联网访问应用分布、互联网访问源IP TOP5、互联网访问目的IP TOP5、主动外联流量趋势、主动外联目的地域分布(中国)、目的地域分布(世界)。

前提条件

- 已采集CFW日志，详情请参见**CFW接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云防火墙（Cloud Firewall，CFW）是新一代的云原生防火墙，提供云上互联网边界和VPC边界的防护，包括实时入侵检测与防御、全局统一访问控制、全流量分析可视化、日志审计与溯源分析等，同时支持按需弹性扩容、AI提升智能防御能力、灵活扩展满足云上业务的变化和扩张需求，极简应用让用户快速灵活应对威胁。云防火墙可以通过攻击事件日志查看检测到的危险流量的危险等级、受影响的端口、命中的规则、攻击事件类型等信息；通过访问控制日志查看根据访问控制策略放行或阻断的所有流量，以便更好的调整访问控制策略。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“CFW仪表盘模板>CFW流量日志中心”仪表盘，查看图表详情。

----结束

重要图表说明

CFW流量日志中心仪表盘中的重要图表说明如下所示：

- **互联网访问流量趋势**图展示互联网访问流量趋势的变化情况，所关联的查询分析语句如下所示：
  ```
  select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, SUM(to_s_bytes) AS '入流量', SUM(to_c_bytes) AS '出流量' WHERE direction='out2in' group by t_time order by t_time
  ```
- **互联网访问流入地域分布(中国)**图展示互联网访问流入地域分布(中国)的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT count(*) AS PV, ip_to_province(src_ip) AS province WHERE direction='out2in' and IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*') ORDER BY PV DESC
  ```
- **互联网访问流入地域分布(世界)**图展示互联网访问流入地域分布(世界)的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT count(*) AS PV, ip_to_country(src_ip) AS country WHERE direction='out2in' GROUP BY country HAVING country not in ('','保留地址','*') ORDER BY PV DESC
  ```
- **互联网访问应用分布**图展示互联网访问应用分布的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT app, COUNT(*) AS num WHERE direction='out2in' GROUP BY app ORDER BY num DESC
  ```
- **互联网访问源IP TOP5**图展示互联网访问源IP TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY src_ip ORDER BY sum_bytes DESC LIMIT 5
  ```
- **互联网访问目的IP TOP5**图展示互联网访问目的IP TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='out2in' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
  ```
- **主动外联流量趋势**图展示主动外联流量趋势的变化情况，所关联的查询分析语句如下所示：


```
select time_series(MILLIS_TO_TIMESTAMP(start_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, SUM(to_c_bytes) AS '入流量', SUM(to_s_bytes) AS '出流量' WHERE direction='in2out' group by t_time order by t_time
```

- **主动外联目的地域分布(中国)**图展示主动外联目的地域分布(中国)的变化情况，所关联的查询分析语句如下所示：

```
SELECT count(*) AS PV, ip_to_province(dst_ip) AS province WHERE direction='in2out' and IP_TO_COUNTRY (dst_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*') ORDER BY PV DESC
```

- **目的地域分布(世界)**图展示目的地域分布(世界)的变化情况，所关联的查询分析语句如下所示：

```
SELECT count(*) AS PV, ip_to_country(dst_ip) AS country WHERE direction='in2out' GROUP BY country HAVING country not in ('','保留地址','*') ORDER BY PV DESC
```

- **主动外联-应用分布**图展示主动外联-应用分布的变化情况，所关联的查询分析语句如下所示：

```
SELECT app, COUNT(*) AS num WHERE direction='in2out' GROUP BY app ORDER BY num DESC
```

- **主动外联源IP TOP5**图展示主动外联源IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
select src_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY src_ip ORDER BY sum_bytes DESC LIMIT 5
```

- **主动外联目的IP TOP5**图展示主动外联目的IP TOP5的变化情况，所关联的查询分析语句如下所示：

```
select dst_ip, SUM(bytes)/1024 as sum_bytes WHERE direction='in2out' GROUP BY dst_ip ORDER BY sum_bytes DESC LIMIT 5
```

7.3.3.5.3 CFW 攻击日志中心

CFW攻击日志中心仪表盘主要展示攻击趋势、攻击来源分布(中国)、攻击来源分布(世界)、攻击类型分布、攻击目的TOP5、攻击来源TOP5。

前提条件

- 已采集CFW日志，详情请参见**CFW接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

云防火墙（Cloud Firewall，CFW）是新一代的云原生防火墙，提供云上互联网边界和VPC边界的防护，包括实时入侵检测与防御、全局统一访问控制、全流量分析可视化、日志审计与溯源分析等，同时支持按需弹性扩容、AI提升智能防御能力、灵活扩展满足云上业务的变化和扩张需求，极简应用让用户快速灵活应对威胁。云防火墙可以通过攻击事件日志查看检测到的危险流量的危险等级、受影响的端口、命中的规则、攻击事件类型等信息；通过访问控制日志查看根据访问控制策略放行或阻断的所有流量，以便更好的调整访问控制策略。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“CFW日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“CFW仪表盘模板>CFW攻击日志中心”仪表盘，查看图表详情。

----结束

重要图表说明

CFW攻击日志中心仪表盘中的重要图表说明如下所示：

- **攻击趋势**图展示攻击趋势的变化情况，所关联的查询分析语句如下所示：
  ```
  select time_series(MILLIS_TO_TIMESTAMP(event_time), 'PT1M', 'yyyy-MM-dd HH:mm:ss', '0') as t_time, count(*) as frequency group by t_time order by t_time
  ```
- **攻击来源分布(中国)**图展示攻击来源分布(中国)的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT count(*) as PV,ip_to_province(src_ip) as province WHERE  IP_TO_COUNTRY (src_ip) = '中国' GROUP BY province HAVING province not in ('','保留地址','*')
  ```
- **攻击来源分布(世界)**图展示攻击来源分布(世界)的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT count(*) AS PV,ip_to_country(src_ip) AS country GROUP BY country HAVING country not in ('','保留地址','*')
  ```
- **攻击类型分布**图展示攻击类型分布的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT attack_type, COUNT(*) as num GROUP BY attack_type ORDER BY num
  ```
- **攻击目的TOP5**图展示攻击目的TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT dst_ip, COUNT(*) as frequency GROUP BY dst_ip ORDER BY frequency  DESC LIMIT 5
  ```
- **攻击来源TOP5**图展示攻击来源TOP5的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT src_ip, COUNT(*) as frequency GROUP BY src_ip ORDER BY frequency DESC LIMIT 5
  ```

7.3.3.6 CSE 仪表盘模板

7.3.3.6.1 CSE 层访问中心

CSE层访问中心仪表盘主要展示互联网访问-拦截趋势、主动外联-拦截趋势、互联网阻断应用TOP5、互联网阻断目的TOP5、互联网阻断来源TOP5、主动外联阻断应用TOP5、主动外联阻断目的TOP5、主动外联阻断来源TOP5。

前提条件

- 已采集CSE日志。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

微服务引擎（Cloud Service Engine，CSE），是用于微服务应用的云中间件，支持华为云贡献到Apache社区的注册配置中心Servicecomb引擎和开源增强的注册配置中心Nacos引擎。用户可结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CSE仪表盘模板>CSE层访问中心”仪表盘，查看图表详情。

----结束

CSE层访问中心仪表盘中的过滤器说明如下所示：

- 过滤上游IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_host)
  ```
- 过滤调用链trace_id，所关联的查询分析语句如下所示：
  ```
  select distinct(trace_id)
  ```

重要图表说明

CSE层访问中心仪表盘中的重要图表说明如下所示：

- **PV对比昨日**图展示PV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
  ```
- **PV对比上周**图展示PV数对比上周的变化情况，所关联的查询分析语句如下所示
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
  ```
- **UV对比昨日**图展示UV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
  ```
- **UV对比上周**图展示UV数对比上周的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(authority) as "uv" from log))
  ```
- **访问量PV分布(中国)**图展示中国区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
  ```
  select ip_to_province(authority) as province, sum(ori_pv) as pv from (select authority, count(1) as ori_pv  group by authority  ORDER BY ori_pv desc  LIMIT 10000)  where IP_TO_COUNTRY (authority) = '中国'  group by province HAVING province not in ('','保留地址','*')
  ```
- **访问量PV分布(世界)**图展示世界区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT ip_to_country(authority) as country,sum(ori_pv) as PV from (select authority, count(1) as ori_pv  group by authority  ORDER BY ori_pv desc  LIMIT 10000) GROUP BY country HAVING country not in ('','保留地址','*')
  ```
- **平均时延分布（中国）**图展示中国区域内平均时延的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)" FROM (SELECT ip_to_province(authority) as province,sum(rt)/sum(ori_pv) * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt,count(1) as ori_pv  group by authority  ORDER BY ori_pv desc  LIMIT 10000) WHERE  IP_TO_COUNTRY (authority) = '中国' GROUP BY province )  where province not in ('','保留地址','*')
  ```
- **平均时延分布（世界）**图展示世界区域内平均时延的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)" FROM (SELECT ip_to_country(authority) as country,sum(rt)/sum(ori_pv)  * 1000 AS "平均延迟(ms)" from (select authority, sum(duration) as rt,count(1) as ori_pv  group by authority  ORDER BY ori_pv desc LIMIT 10000) GROUP BY country ) where  country not in ('','保留地址','*')
  ```
- **今日PV/UV**图展示今日PV/UV数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(authority) as UV from log WHERE __time <= CURRENT_TIMESTAMP  and __time >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_)
  ```
- **区域访问TOP10(省份)**图展示访问排名前十的省份，所关联的查询分析语句如下所示：
  ```
  select ip_to_province(authority) as "province", sum(ori_pv) as "访问次数" from (select authority, count(1) as ori_pv  group by authority  ORDER BY ori_pv desc  LIMIT 10000) group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
  ```
- **区域访问TOP10(城市)**图展示访问排名前十的城市，所关联的查询分析语句如下所示：


```
select ip_to_city(authority) as "city", sum(ori_pv) as "访问次数" from (select authority, count(1) as ori_pv  group by authority ORDER BY ori_pv desc LIMIT 10000) group by "city" HAVING  "city" <> '-1' order by "访问次数" desc  limit 10
```

- **Host访问TOP10**图展示访问排名前十的主机，所关联的查询分析语句如下所示：
```
select  upstream_host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10**图展示访问排名前十的UserAgent，所关联的查询分析语句如下所示：
```
select user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)**图展示各终端设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by  type
```
- **设备占比(系统)**图展示各系统设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by  type HAVING type != 'other'
```
- **TOP URL**图展示访问前十url的PV、UV及访问成功率等信息，所关联的查询分析语句如下所示：
```
select path , count(1) as pv, APPROX_COUNT_DISTINCT(authority) as UV, round(sum( case when response_code < 400 then 1 else 0 end  ) * 100.0 / count(1), 2) as "访问成功率" group by path ORDER by pv desc
```
- **TOP 访问IP**图展示访问前十的IP及城市、运营商和PV等数据，所关联的查询分析语句如下所示：
```
select authority as "来源IP",ip_to_country(authority) as "国家",ip_to_province(authority) as "省份",ip_to_city(authority) as "城市",ip_to_provider(authority) as "运营商",count(1) as "PV" group by authority ORDER by "PV" desc limit 100
```

7.3.3.6.2 CSE 层监控中心

CSE层监控中心仪表盘主要展示互联网访问流量趋势、互联网访问流入地域分布(中国)、互联网访问流入地域分布(世界)、互联网访问应用分布、互联网访问源IP TOP5、互联网访问目的IP TOP5、主动外联流量趋势、主动外联目的地域分布(中国)、目的地域分布(世界)。

前提条件

- 已采集CSE日志。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

微服务引擎（Cloud Service Engine，CSE），是用于微服务应用的云中间件，支持华为云贡献到Apache社区的注册配置中心Servicecomb引擎和开源增强的注册配置中心Nacos引擎。用户可结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CSE仪表盘模板>CSE层监控中心”仪表盘，查看图表详情。

----结束

CSE层监控中心仪表盘中的过滤器说明如下所示：

- 过滤上游IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_host)
  ```
- 过滤调用链trace_id，所关联的查询分析语句如下所示：
  ```
  select distinct(trace_id)
  ```

重要图表说明

CSE层监控中心仪表盘中的重要图表说明如下所示：

- **访问量PV**图展示访问量的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,PV FROM ( SELECT TIME_CEIL ( TIME_PARSE(start_time), 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log GROUP BY _time_ )
  ```
- **请求成功率**图展示请求成功率的变化情况，所关联的查询分析语句如下所示：
  ```
  select ROUND(sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
  ```
- **平均延迟**图展示平均延迟的变化情况，所关联的查询分析语句如下所示：
  ```
  select round(avg(duration) * 1000, 3) as cnt
  ```
- **4XX请求数**图展示4xx的请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "response_code" >= 400 and "response_code" < 500
  ```
- **404请求数**图展示404请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "response_code" = 404
  ```
- **429请求数**图展示429请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "response_code" = 429
  ```
- **504请求数**图展示504请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "response_code" = 504
  ```
- **5XX请求数**图展示5xx的请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,cnt FROM ( SELECT TIME_CEIL ( TIME_PARSE(start_time), 'PT300S' ) AS _time_, count( 1 ) AS cnt FROM log where "response_code" >= 500 GROUP BY _time_ )
  ```
- **状态码分布**图展示请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT response_code, COUNT(1) AS rm GROUP BY response_code
  ```
- **访问量UV**图展示访问量UV的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,UV FROM (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(authority) as UV  from log group by _time_)
  ```
- **流量**图展示入流量和出流量的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss') AS _time_,round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量",round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_,sum(bytes_received) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by _time_)
  ```
- **访问失败率**图展示访问失败率和5xx的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') AS _time_,sum(case when response_code >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when response_code >=500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by _time_)
  ```
- **延迟**图展示访问P50、P90、P99、P9999延迟的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss') as _time_,round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE WHEN
  ```



```
"P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(start_time),'PT600S') as _time_,avg(duration) * 1000 as "平均", APPROX_QUANTILE_DS("duration", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("duration", 0.90)*1000 as "P90" ,APPROX_QUANTILE_DS("duration", 0.99)*1000 as 'P99',APPROX_QUANTILE_DS("duration", 0.9999)*1000 as 'P9999' group by _time_)
```

- **Host请求TOP**图展示主机请求TOP信息的变化情况，所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != '' GROUP BY "upstream_host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图展示主机延迟TOP信息的变化情况，所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != '' GROUP BY "upstream_host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图展示主机访问失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT "upstream_host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "upstream_host", count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != '' GROUP BY "upstream_host") ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图展示URL请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT path, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)" FROM ( SELECT path, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "upstream_host" != '' GROUP BY path ) ORDER BY pv desc
```

- **URL失败率TOP**图展示URL失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT path, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != '' GROUP BY path ) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图展示后端请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入
```



```
流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT authority as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( authority ) AS uv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", sum( bytes_received ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "upstream_host" != ''  GROUP BY addr  having length(authority) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图展示后端延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT authority as addr,count( 1 ) AS pv,sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( duration ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != '' and "authority" != '-' GROUP BY addr ) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图展示后端失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT authority as addr, count( 1 ) AS pv, sum( CASE WHEN "response_code" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != '' and "authority" != '-' GROUP BY addr) ORDER BY "访问失败率(%)" desc
```

- **URL延迟TOP**图展示URL延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT path, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT path, count( 1 ) AS pv, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( duration ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(duration, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(duration, 0.99) * 1000 AS "P99延迟(ms)" WHERE "upstream_host" != ''  GROUP BY path  ) ORDER BY "平均延迟(ms)" desc
```

7.3.3.6.3 CSE 层秒级监控

CSE层秒级监控仪表盘主要展示Upstream状态码等信息，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

- 已采集CSE日志。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

微服务引擎（Cloud Service Engine，CSE），是用于微服务应用的云中间件，支持华为云贡献到Apache社区的注册配置中心Servicecomb引擎和开源增强的注册配置中心Nacos引擎。用户可结合其他云服务，快速构建云原生微服务体系，实现微服务应用的快速开发和高可用运维。高

分析CSE层秒级监控情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“CSE仪表盘模板>CSE层秒级监控”仪表盘，查看图表详情。

----结束

CSE层秒级监控仪表盘中的过滤器说明如下所示：

- 过滤上游IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_host)
  ```
- 过滤调用链trace_id，所关联的查询分析语句如下所示：
  ```
  select distinct(trace_id)
  ```

重要图表说明

CSE层秒级监控仪表盘中的重要图表说明如下所示：

- **QPS**图展示QPS的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') AS _time_ , COUNT(*) as QPS from log group by _time_
  ```
- **成功率**图展示成功率的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time, sum(case when response_code < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
  ```
- **延迟**图展示访问延时的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time, avg(duration)* 1000 as '访问延迟',avg(upstream_service_time)* 1000 as 'Upstream延迟' from log group by __time)
  ```
- **流量**图展示请求流量和返回body流量的变化情况，所关联的查询分析语句如下所示：
  ```
  select __time,round( CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 3 ) AS "请求流量",round( CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 3 ) AS "返回body流量" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(start_time),'PT5S'),'yyyy-MM-dd HH:mm:ss') as __time , sum("bytes_received") / 1024.0 as "请求流量", sum("bytes_sent") / 1024.0 as "返回body流量" group by __time)
  ```
- **状态码**图展示响应状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_CEIL ( TIME_PARSE ( start_time ), 'PT5S' ) AS "time", SUM( CASE WHEN "response_code" >= 200 AND "response_code" < 300 THEN 1 ELSE 0 END ) AS "2XX", SUM( CASE WHEN "response_code" >= 300 AND "response_code" < 400 THEN 1 ELSE 0 END ) AS "3XX", SUM( CASE WHEN "response_code" >= 400 AND "response_code" < 500 THEN 1 ELSE 0 END ) AS "4XX", SUM( CASE WHEN "response_code" >= 500 AND "response_code" < 600 THEN 1 ELSE 0 END ) AS "5XX", SUM( CASE WHEN "response_code" < 200 OR "response_code" >= 600 THEN 1 ELSE 0 END ) AS "其他" FROM log  WHERE TIME_PARSE ( start_time ) IS NOT NULL GROUP BY "time" ORDER BY "time" ASC LIMIT 100000
  ```

7.3.3.7 CTS 日志中心

云审计服务（Cloud Trace Service，简称CTS）日志中记录了云服务管理操作事件，可以帮助您实现安全合规分析。

说明

CTS暂不支持仪表盘模板。

接入日志

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“CTS日志中心”，选择“接入日志”，进入接入日志页面。

步骤3 **配置CTS日志接入**。

----结束

查看文档

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“CTS日志中心”，选择“查看文档”，进入对应的华为云帮助文档。

步骤3 参考帮助文档，进行CTS接入配置。

----结束

7.3.3.8 DCS 仪表盘模板

云日志服务支持日志采集向导一站式采集DCS日志，并为DCS日志配置结构化和DCS审计日志中心仪表盘。

前提条件

日志配置结构化，详情请参见**结构化配置**。

背景信息

分布式缓存服务（Distributed Cache Service，简称DCS）是华为云提供的一款内存数据库服务，兼容了Redis内存数据库引擎，满足用户高并发及数据快速访问的业务诉求。云日志服务提供对各种云资源操作记录的收集、存储和查询功能，可用于支撑安全分析、合规审计、资源跟踪和问题定位等常见应用场景。

DCS 审计日志中心

DCS审计日志中心仪表盘主要展示访问用户数、访问客户端数、审计日志条数、平均响应时间、平均QPS、TOP5 用户、TOP5 客户端、TOP5 执行命令、热Key、审计日志详情。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“DCS仪表盘模板>DCS审计日志中心”仪表盘，查看图表详情。

----结束

重要图表说明

DCS审计日志中心仪表盘中的重要图表说明如下所示：

- **访问用户数**图展示访问用户个数，所关联的查询分析语句如下所示：
  ```
  select count(distinct(user)) as user_num
  ```
- **访问客户端数**图展示访问客户端个数，所关联的查询分析语句如下所示：


```
select count(distinct(client_addr)) as client_num
```

- **审计日志条数**图展示审计日志条数，所关联的查询分析语句如下所示：
```
select count(1) as log_num
```
- **平均响应时间**图展示平均响应时间，所关联的查询分析语句如下所示：
```
select avg(use_time) as avg_time
```
- **平均QPS**图展示QPS数量，所关联的查询分析语句如下所示：
```
select count(*) / CAST((TIMESTAMPDIFF (minute, MIN(__time),MAX(__time))+1) as FLOAT)
```
- **TOP5 用户**图展示TOP5 用户情况，所关联的查询分析语句如下所示：
```
select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5
```
- **TOP5 客户端**图展示TOP5 客户端数量情况，所关联的查询分析语句如下所示：
```
select client_addr, count(1) as 'remote_count' group by client_addr order by count(1) desc limit 5
```
- **TOP5 执行命令**图展示TOP5 执行命令的变化情况，所关联的查询分析语句如下所示：
```
select command_name, count(1) as 'command_count' group by command_name order by count(1) desc
```
- **热Key**图展示Key数量的变化情况，所关联的查询分析语句如下所示：
```
SELECT count(*) as 'count',  command_keys GROUP by command_keys order by count(*) desc limit 5
```
- **审计日志详情**图展示审计日志执行时间、客户端IP、客户端类型等详情，所关联的查询分析语句如下所示：
```
select  "client_addr" as '客户端IP',"client_type" as '客户端类型',"server_addr" as '服务端IP',"command_name" as '执行命令',"command_keys" as '命令KEYS',"command_param" as '命令内容',"command_type" as '命令类型',"use_time" as '执行耗时',"time" as '执行时间',"db" as 'DB',"user" as '账号名',"instance_id" as '实例ID',"role" as '节点角色',"extend" as '扩展信息'
```

7.3.3.9 DDS 仪表盘模板

云日志服务支持日志采集向导一站式采集DDS日志，并为DDS日志配置结构化和DDS审计日志中心仪表盘。

前提条件

- 已采集DDS日志，详情请参见**DDS接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

文档数据库服务（Document Database Service）完全兼容MongoDB协议，提供安全、高可用、高可靠、弹性伸缩和易用的数据库服务，同时提供一键部署、弹性扩容、容灾、备份、恢复、监控和告警等功能。云日志服务进行分析日志、搜索日志、日志可视化、下载日志和查看实时日志等操作。

DDS 审计日志中心

DDS审计日志中心仪表盘主要展示操作类型、客户端IP、数据库名、用户名、集合名、审计日志条数、访问用户数、访问客户端数、TOP5 执行命令、TOP5 用户、TOP5 客户端、审计日志详情。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“DDS仪表盘模板>DDS审计日志中心”仪表盘，查看图表详情。

----结束

DDS审计日志中心仪表盘中的过滤器说明如下所示：

- 操作类型，所关联的查询分析语句如下所示：
  ```
  select distinct(optype)
  ```
- 客户端IP，所关联的查询分析语句如下所示：
  ```
  select distinct(user_ip)
  ```
- 数据库名，所关联的查询分析语句如下所示：
  ```
  select distinct(db)
  ```
- 用户名，所关联的查询分析语句如下所示：
  ```
  select distinct(user)
  ```
- 集合名，所关联的查询分析语句如下所示：
  ```
  select distinct(coll)
  ```

重要图表说明

DDS审计日志中心仪表盘中的重要图表说明如下所示：

- **审计日志条数**图展示审计日志条数的变化情况，所关联的查询分析语句如下所示：
  ```
  select count(1) as log_num
  ```
- **访问用户数**图展示访问用户数的变化情况，所关联的查询分析语句如下所示：
  ```
  select count(distinct(user)) as user_num
  ```
- **访问客户端数**图展示访问客户端数的变化情况，所关联的查询分析语句如下所示：
  ```
  select count(distinct(user_ip)) as client_num
  ```
- **TOP5 执行命令**图展示TOP5 执行命令的变化情况，所关联的查询分析语句如下所示：
  ```
  select optype, count(1) as 'command_count' group by optype order by count(1) desc LIMIT 5
  ```
- **TOP5 用户**图展示TOP5 用户的变化情况，所关联的查询分析语句如下所示：
  ```
  select user, count(1) as 'user_count' group by user order by count(1) desc LIMIT 5
  ```
- **TOP5 客户端**图展示TOP5 客户端的变化情况，所关联的查询分析语句如下所示：
  ```
  select user_ip, count(1) as 'remote_count' group by user_ip order by count(1) desc LIMIT 5
  ```
- **审计日志详情**图展示审计日志详情的变化情况，所关联的查询分析语句如下所示：
  ```
  select "time" as "执行时间","user" as "用户名","param" as "查询语句","instanceid" as "实例ID","db" as "DB","coll" as "集合名","user_ip" as "客户端IP"
  ```

7.3.3.10 DMS 仪表盘模板

云日志服务支持日志采集向导一站式采集DMS重平衡日志，支持多维度分析，并为DMS-Rebalance日志配置结构化和仪表盘。该仪表盘主要展示DMS重平衡日志的重平衡消费组个数、重平衡次数、消费组重平衡次数、重平衡原因及组详情。

前提条件

- 已采集DMS日志，详情请参见**CFW接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

分布式消息服务Kafka版（Distributed Message Service for Kafka）是一款基于开源社区版Kafka提供的消息队列服务，向用户提供计算、存储和带宽资源独占式的Kafka专享实例。重平衡日志记录Rebalance的详情，包括Rebalance时间、原因和触发Rebalance的客户端等。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“DMS仪表盘模板>DMS重平衡日志中心”仪表盘，查看图表详情。

----结束

DMS重平衡日志中心仪表盘中的过滤器说明如下所示：

- 消费组ID，所关联的查询分析语句如下所示：
  ```
  select distinct("message.groupId")
  ```

重要图表说明

DMS重平衡日志中心日志中心仪表盘中的重要图表说明如下所示：

- 重平衡消费组个数图展示当触发Rebalance的操作非RESPONSE和REQUEST时的不同消费组ID个数，所关联的查询分析语句如下所示：
  ```
  select count(distinct("message.groupId"))  as "total" from log where ("message.type"!='RESPONSE' and "message.type" !='REQUEST')
  ```
- 重平衡次数图展示当触发Rebalance的操作非RESPONSE和REQUEST时的次数，所关联的查询分析语句如下所示：
  ```
  select count(*) as 'total'  where ("message.type" !='RESPONSE' and "message.type" !='REQUEST')
  ```
- 消费组重平衡次数图展示当触发Rebalance的操作非RESPONSE和REQUEST时的不同消费组的个数，所关联的查询分析语句如下所示：
  ```
  select "message.groupId" as 'GroupId', count(*) as 'Count' where ('message.type' !='RESPONSE' and 'message.type' !='REQUEST')  group by "message.groupId"
  ```
- 重平衡原因及组详情图展示发生重平衡时的原因和组详情，所关联的查询分析语句如下所示：
  ```
  select __time  as 'Time', "message.type" as 'Type' ,"message.groupId" as 'GroupId', "message.reason" as 'Reason',"message.group" as 'Group' where ('message.type' !='RESPONSE' and 'message.type' !='REQUEST')
  ```

7.3.3.11 DSL 仪表盘模板

云日志服务支持DSL仪表盘模板，DSL加工任务监控中心主要展示加工任务ID、加工任务名称、输入行数、输出行数等信息。

前提条件

- 已创建DSL加工任务。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

DSL（Domain Specific Language）加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘”。

步骤2 在仪表盘模板下方，选择“DSL仪表盘模板>DSL加工任务监控中心”仪表盘，查看图表详情。

----结束

DSL加工任务监控中心仪表盘中的过滤器说明如下所示：

- 获取加工任务ID，所关联的查询分析语句如下所示：
```
select distinct(task_id)
```
- 获取加工任务名称，所关联的查询分析语句如下所示：
```
select distinct(task_name)
```

DSL加工任务监控中心表盘中的重要图表说明如下所示：

- **输入行数**图展示输入行数的变化情况，所关联的查询分析语句如下所示：
```
SELECT CASE  WHEN  "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' )  WHEN "input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "input" < 1000000000 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "input"/ 1000.0 < 1000000000 THEN concat( cast( round( "input"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' )  END AS  "total"  from (select sum("process.accept") as "input")
```
- **输出行数**图展示输出行数的变化情况，所关联的查询分析语句如下所示
```
SELECT CASE  WHEN  "delivered" < 1000 THEN concat( cast( "delivered" AS VARCHAR ), '行' ) WHEN "delivered" < 1000 * 1000 THEN concat( cast( round( "delivered"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "delivered" < 1000000000 THEN concat( cast( round( "delivered"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  WHEN "delivered"/ 1000.0 < 1000000000 THEN
concat( cast( round( "delivered"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE
concat( cast( round( "delivered"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' )  END AS  "total"  from (select sum("process.delivered") as "delivered")
```
- **过滤行数**图展示满足过滤条件行数的变化情况，所关联的查询分析语句如下所示：
```
SELECT CASE  WHEN  "drop" < 1000 THEN concat( cast( "drop" AS VARCHAR ), '行' )  WHEN "drop" < 1000 * 1000 THEN concat( cast( round( "drop"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "drop" < 1000000000 THEN concat( cast( round( "drop"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' )  WHEN "drop"/ 1000.0 < 1000000000 THEN concat( cast( round( "drop"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "drop"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' )  END AS  "total"  from (select sum("process.drop") as "drop")
```
- **失败行数**图展示失败行数的变化情况，所关联的查询分析语句如下所示：
```
SELECT CASE  WHEN  "failed" < 1000 THEN concat( cast( "failed" AS VARCHAR ), '行' )  WHEN "failed" < 1000 * 1000 THEN concat( cast( round( "failed"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "failed" < 1000000000 THEN concat( cast( round( "failed"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "failed"/ 1000.0 < 1000000000 THEN concat( cast( round( "failed"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "failed"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' )  END AS  "total"  from (select sum("process.failed") as "failed")
```
- **执行记录**图展示执行记录的分布情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT( MILLIS_TO_TIMESTAMP("start"), 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00') as "统计开始时间",TIME_FORMAT( MILLIS_TO_TIMESTAMP("end"), 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00') as "统计结束时间", "process.accept" as "输入行数", "process.delivered" as "输出行数", "process.drop" as "过滤行数", "process.failed" as "失败行数" limit 1000
```

7.3.3.12 ELB 仪表盘模板

云日志服务联合华为云弹性负载均衡（ Elastic Load Balance ）提供开箱即用仪表盘模板，帮助您了解用户行为、地域分布、请求成功率和响应延迟等信息。支持日志采集向导一站式采集ELB日志和将ELB日志结构化。仪表盘包括ELB7层秒级监控、ELB7层监控中心和ELB7层访问中心。

7.3.3.12.1 ELB7 层访问中心

ELB7层访问中心仪表盘主要展示PV对比昨日、PV对比上周、UV对比昨日、UV对比上周、访问量PV分布(中国)、访问量PV分布(世界)、访问量UV分布(中国)、访问量UV分

布(世界)、平均时延分布（中国）、平均时延分布（世界）、今日PV/UV、7日PV/UV、区域访问TOP10(省份)、区域访问TOP10(城市)、Host访问TOP10、UserAgent访问TOP10、设备占比(终端)、设备占比(系统)、TOP URL、TOP 访问IP。

前提条件

- 已采集ELB日志，详情请参见**ELB接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“ELB仪表盘模板>ELB7层访问中心”仪表盘，查看图表详情。

----结束

ELB7层访问中心仪表盘中的过滤器说明如下所示：

- 获取所有负载均衡器，所关联的查询分析语句如下所示：
  ```
  select distinct(lb_name)
  ```
- 获取所有客户端IP，所关联的查询分析语句如下所示：
  ```
  select distinct(remote_addr)
  ```
- 获取所有后端服务器IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_addr)
  ```
- 获取所有弹性IP地址，所关联的查询分析语句如下所示：
  ```
  select distinct(eip_address)
  ```

重要图表说明

ELB7层访问中心仪表盘中的重要图表说明如下所示：

说明

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **PV对比昨日**图展示PV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
  ```


- **PV对比上周**图展示PV对比上周的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
```
- **UV对比昨日**图展示UV对比昨日的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```
- **UV对比上周**图展示UV对比上周的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(remote_addr) as "uv" from log))
```
- **访问量PV分布(中国)**图展示访问量PV分布(中国)的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国'  group by province HAVING province not in ('','保留地址','*')
```
- **访问量PV分布(世界)**图展示访问量PV分布(世界)的变化情况，所关联的查询分析语句如下所示：
```
SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
```
- **访问量UV分布(中国)**图展示访问量UV分布(中国)的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as uv where IP_TO_COUNTRY (remote_addr) = '中国' group by province HAVING province not in ('','保留地址','*')
```
- **访问量UV分布(世界)**图展示访问量UV分布(世界)的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','*')
```
- **平均时延分布（中国）**图展示平均时延分布（中国）的变化情况，所关联的查询分析语句如下所示：
```
SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE  IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址','*'))
```
- **平均时延分布（世界）**图展示平均时延分布（世界）的变化情况，所关联的查询分析语句如下所示：
```
SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
```
- **今日PV/UV**图展示今日PV/UV的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_ , count(1) as PV,  APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ') <= CURRENT_TIMESTAMP  and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_)
```
- **7日PV/UV**图展示7日PV/UV的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_ , count(1) as PV,  APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ') <= CURRENT_TIMESTAMP  and TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ') >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_ )
```
- **区域访问TOP10(省份)**图展示区域访问TOP10(省份)的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province" HAVING "province" <> '-1' order by "访问次数" desc limit 10
```


- **区域访问TOP10(城市)**图展示区域访问TOP10(城市)的变化情况，所关联的查询分析语句如下所示：
```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING  "city" <> '-1' order by "访问次数" desc  limit 10
```
- **Host访问TOP10**图展示Host访问TOP10的变化情况，所关联的查询分析语句如下所示：
```
select  host as "Host", count(1) as "PV" group by "Host" order by "PV" desc limit 10
```
- **UserAgent访问TOP10**图展示UserAgent访问TOP10的变化情况，所关联的查询分析语句如下所示：
```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" desc limit 10
```
- **设备占比(终端)**图展示设备占比(终端)的变化情况，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by  type
```
- **设备占比(系统)**图展示设备占比(系统)的变化情况，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by  type HAVING type != 'other'
```
- **TOP URL**图展示TOP URL的变化情况，所关联的查询分析语句如下所示：
```
select router_request_uri , count(1) as pv, APPROX_COUNT_DISTINCT(remote_addr)  as uv, round(sum( case when status < 400 then 1 else 0 end  ) * 100.0 / count(1), 2) as "访问成功率" group by router_request_uri ORDER by pv desc
```
- **TOP 访问IP**图展示TOP 访问IP的变化情况，所关联的查询分析语句如下所示：
```
select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as "PV",http_user_agent as "UserAgent采样",router_request_uri as "URL采样" group by remote_addr,http_user_agent,router_request_uri ORDER by "PV" desc
```

7.3.3.12.2 ELB7 层监控中心

ELB7层监控中心仪表盘主要展示访问量PV、请求成功率、4XX请求数、499请求数、平均延迟、404请求数、504请求数、5XX请求数、状态码分布、访问量UV、流量、访问失败率、延迟、Host请求TOP、Host延迟TOP、Host失败率TOP、URL请求TOP、URL延迟TOP、URL失败率TOP、后端请求TOP、后端延迟TOP、后端失败率TOP。

前提条件

- 已采集ELB日志，详情请参见**ELB接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“ELB仪表盘模板>ELB7层监控中心”仪表盘，查看图表详情。

----结束

ELB7层监控中心仪表盘中的过滤器说明如下所示：

- 获取所有负载均衡器，所关联的查询分析语句如下所示：
  ```
  select distinct(lb_name)
  ```
- 获取所有客户端IP，所关联的查询分析语句如下所示：
  ```
  select distinct(remote_addr)
  ```
- 获取所有后端服务器IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_addr)
  ```
- 获取所有弹性IP地址，所关联的查询分析语句如下所示：
  ```
  select distinct(eip_address)
  ```

重要图表说明

ELB7层监控中心仪表盘中的重要图表说明如下所示：

说明

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **访问量PV**图展示访问量PV的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_ , count(1) as PV from log group by _time_ order by _time_ )
  ```
- **请求成功率**图展示请求成功率的变化情况，所关联的查询分析语句如下所示：
  ```
  select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
  ```
- **4XX请求数**图展示4XX请求数的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
  ```
- **499请求数**图展示499请求数的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 499
  ```
- **平均延迟**图展示平均延迟的变化情况，所关联的查询分析语句如下所示：
  ```
  select round(avg(request_time) * 1000, 3) as cnt
  ```
- **404请求数**图展示404请求数的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 404
  ```
- **504请求数**图展示504请求数的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 504
  ```
- **5XX请求数**图展示5XX请求数的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,cnt FROM ( SELECT TIME_CEIL ( TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT300S' ) AS _time_, count( 1 ) AS cnt FROM log where "status" >= 500 GROUP BY _time_ )
  ```
- **状态码分布**图展示状态码分布的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT status, COUNT(1) AS rm GROUP BY status
  ```
- **访问量UV**图展示访问量UV的变化情况，所关联的查询分析语句如下所示：


```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,UV FROM (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV  from log group by _time_)
```

- **流量**图展示流量的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_,round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量",round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_,sum(request_length) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by  _time_)
```

- **访问失败率**图展示访问失败率、5XX比例的变化情况，所关联的查询分析语句如下所示：

```
SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') AS _time_,sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >=500 THEN 1 ELSE 0 END)*100.0/ COUNT(1) as '5XX比例' group by  _time_)
```

- **延迟**图展示延迟的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT600S') as _time_,avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as "P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as "P90" ,APPROX_QUANTILE_DS("request_time", 0.99)*1000 as 'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by  _time_)
```

- **Host请求TOP**图展示Host请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT host, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY "host" ) ORDER BY pv DESC
```

- **Host延迟TOP**图展示Host延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY "host" ) ORDER BY "平均延迟(ms)" desc
```

- **Host失败率TOP**图展示Host失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY "host"  ) ORDER BY "访问失败率(%)" desc
```

- **URL请求TOP**图展示URL请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)"
```



```
ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT router_request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != ''  GROUP BY router_request_uri  ) ORDER BY pv desc
```

- **URL延迟TOP**图展示URL延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY router_request_uri  ) ORDER BY "平均延迟(ms)" desc
```

- **URL失败率TOP**图展示URL失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT router_request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM( SELECT router_request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' GROUP BY router_request_uri) ORDER BY "访问失败率(%)" desc
```

- **后端请求TOP**图展示后端请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( http_x_forwarded_for ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( body_bytes_sent ) / 1024.0 AS "出流量(KB)" WHERE "host" != ''  GROUP BY addr  having length(upstream_addr_priv) > 2) ORDER BY "pv" desc
```

- **后端延迟TOP**图展示后端延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT upstream_addr_priv as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "平均延迟(ms)" desc
```

- **后端失败率TOP**图展示后端失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT upstream_addr_priv as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY addr having length(upstream_addr_priv) > 2) ORDER BY "访问失败率(%)" desc
```


7.3.3.12.3 ELB7 层秒级监控

ELB7层秒级监控仪表盘主要展示QPS、成功率、延迟、流量、状态码、后端响应码。

前提条件

- 已采集ELB日志，详情请参见**ELB接入**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

弹性负载均衡（Elastic Load Balance，简称ELB）是将访问流量根据分配策略分发到后端多台服务器的流量分发控制服务。弹性负载均衡可以通过流量分发扩展应用系统对外的服务能力，同时通过消除单点故障提升应用系统的可用性，支持查看和分析对七层负载均衡HTTP和HTTPS进行请求的详细访问日志记录，包括请求时间、客户端IP地址、请求路径和服务器响应等。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“ELB日志中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“ELB仪表盘模板>ELB7层秒级监控”仪表盘，查看图表详情。

----结束

ELB7层秒级监控仪表盘中的过滤器说明如下所示：

- 获取所有负载均衡器，所关联的查询分析语句如下所示：
  ```
  select distinct(lb_name)
  ```
- 获取所有客户端IP，所关联的查询分析语句如下所示：
  ```
  select distinct(remote_addr)
  ```
- 获取所有后端服务器IP，所关联的查询分析语句如下所示：
  ```
  select distinct(upstream_addr)
  ```
- 获取所有弹性IP地址，所关联的查询分析语句如下所示：
  ```
  select distinct(eip_address)
  ```

重要图表说明

ELB7层秒级监控仪表盘中的重要图表说明如下所示：

说明

在搜索框上方选择时间范围，建议选择相对时间30分钟以上的查询时间（即日志上报时间）。由于日志的业务时间（字段名称为time_iso8601）和日志上报时间的差异，在查询时间范围限制下根据业务时间排序的折线图两端的数据不具有参考性。

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

- **QPS**图展示QPS的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_ , COUNT(*) as QPS from log group by _time_
  ```


- **成功率**图展示成功率的变化情况，所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```

- **延迟**图展示延迟的变化情况，所关联的查询分析语句如下所示：

```
select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from log group by __time)
```

- **流量**图展示流量的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(__time, 'yyyy-MM-dd HH:mm:ss', '+08:00') AS __time, round(CASE WHEN "请求流量" > 0 THEN "请求流量" ELSE 0 END, 2) AS "请求流量",round(CASE WHEN "返回body流量" > 0 THEN "返回body流量" ELSE 0 END, 2) AS "返回body流量" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S') as __time, sum("request_length") / 1024.0 as "请求流量", sum("body_bytes_sent") / 1024.0 as "返回body流量" group by __time )
```

- **状态码**图展示状态码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time", CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX", CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX", CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX", CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX", CASE WHEN e."其他" IS NOT NULL THEN CAST(e."其他" AS BIGINT) ELSE 0 END as "其他" from ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t from log group by t order by t asc ) t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t, CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >= 200 and "status" < 300 group by t order by t asc ) a on t.t = a.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t, CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >= 300 and "status" < 400 group by t order by t asc ) b on t.t = b.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t, CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >= 400 and "status" < 500 group by t order by t asc ) c on t.t = c.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t, CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >= 500 and "status" < 600 group by t order by t asc ) d on t.t = d.t left join ( select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'), 'PT5S' ) as t, CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" < 200 or "status" >= 600 group by t order by t asc ) e on t.t = e.t
```

- **后端响应码**图展示后端响应码的变化情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(t.t, 'yyyy-MM-dd HH:mm:ss', '+08:00') as "time",CASE WHEN a."2XX" IS NOT NULL THEN CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",CASE WHEN b."3XX" IS NOT NULL THEN CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",CASE WHEN c."4XX" IS NOT NULL THEN CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",CASE WHEN d."5XX" IS NOT NULL THEN CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",CASE WHEN e."其他" IS NOT NULL THEN e."其他" ELSE 0 END as "其他" from (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S') as t from log group by t order by t asc) t left join(select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc) a on t.t = a.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b on t.t =b.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc) c on t.t =c.t left join (select TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd''T''HH:mm:ssZZ'),'PT5S') as t ,
```



```
CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and
"upstream_status" < 600 group by t order by t asc) d on t.t =d.t left join (select
TIME_CEIL(TIME_PARSE(SUBSTRING(time_iso8601, 2, 25) ,'yyyy-MM-dd"T"HH:mm:ssZZ'),'PT5S') as t ,
CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "upstream_status" < 200 or
"upstream_status" >= 600 group by t order by t asc) e on t.t =e.t
```

7.3.3.13 ER 仪表盘模板

云日志服务支持日志采集向导一站式采集ER日志，支持多维度分析，并为ER日志配置结构化和仪表盘。该仪表盘主要展示ER日志的TOP20包数统计、TOP20流量统计、流日志条数、流日志详情，并且可以通过实例ID、连接ID、流量方向、源IP、目的IP、协议类型进行过滤。

前提条件

- 已采集ER流量日志，详情请参见**企业路由器ER接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

企业路由器（Enterprise Router, ER）可以连接虚拟私有云（Virtual Private Cloud, VPC）或本地网络来构建中心辐射型组网，是云上大规格、高带宽、高性能的集中路由器。企业路由器使用边界网关协议（Border Gateway Protocol, BGP），支持路由学习、动态选路以及链路切换，极大的提升网络的可扩展性及运维效率，从而保证业务的连续性。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“ER仪表盘模板>ER流量日志中心”仪表盘，查看图表详情。

----结束

ER流量日志中心仪表盘中的过滤器说明如下所示：

- 实例ID图，所关联的查询分析语句如下所示：
  ```
  SELECT DISTINCT(instance_id)
  ```
- 连接ID图，所关联的查询分析语句如下所示：
  ```
  SELECT DISTINCT(resource_id)
  ```
- 流量方向图展示为静态过滤器，可按照入方向和出方向的流量进行过滤，所关联的查询分析语句如下所示：
- 源IP图，所关联的查询分析语句如下所示：
  ```
  SELECT DISTINCT(srcaddr)
  ```
- 目的IP图，所关联的查询分析语句如下所示：
  ```
  SELECT DISTINCT(dstaddr)
  ```
- 协议类型图，所关联的查询分析语句如下所示：
  ```
  SELECT DISTINCT(protocol)
  ```

ER流量日志中心仪表盘中的重要图表说明如下所示：

- TOP20包数统计图展示按照包数排名，其中包括源地址，目的地址，包数，连接ID和实例ID详情前20，所关联的查询分析语句如下所示：


```
SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("packets") as "包数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "包数" desc limit 20
```

- TOP20流量统计图展示按照字节数排名，其中包括源地址，目的地址，字节数，连接ID和实例ID详情前20，所关联的查询分析语句如下所示：

```
SELECT "srcaddr" as "源地址", "dstaddr" as "目的地址", sum("bytes") as "字节数", "resource_id" as "连接ID", "instance_id" as "实例ID" group by "instance_id", "resource_id", "srcaddr", "dstaddr" order by "字节数" desc limit 20
```

- 流日志条数图展示每小时流日志条数的数量，所关联的查询分析语句如下所示：

```
select time_series(__time, 'PT1H', 'yyyy-MM-dd HH:mm:ss', '0', '+08:00') as "时间", count(*) as "流日志条数" group by "时间" order by "时间"
```

- 流日志详情图展示日志详情信息，所关联的查询分析语句如下所示：

```
SELECT "instance_id" as "实例ID", "resource_id" as "连接ID", "project_id" as "项目ID", "srcaddr" as "源IP", "dstaddr" as "目的IP", "srcport" as "源端口", "dstport" as "目的端口", "protocol" as "协议类型", "direct" as "流量方向", "packets" as "包数", "bytes" as "字节数",
TIME_FORMAT( MILLIS_TO_TIMESTAMP("start"*1000), 'yyyy-MM-dd HH:mm:ss', '+08:00') as "开始时间", TIME_FORMAT( MILLIS_TO_TIMESTAMP("end"*1000) , 'yyyy-MM-dd HH:mm:ss', '+08:00') as "结束时间"
```

7.3.3.14 METRIC 仪表盘模板

7.3.3.14.1 日志生成指标任务监控中心

云日志服务支持METRIC仪表盘模板，日志生成指标任务监控中心主要展示输入行数、输出行数、满足过滤条件行数、不满足过滤条件行数等信息。

前提条件

- 已创建日志生成指标，详情请参见**日志生成指标（邀测）**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到aom的Prometheus实例，操作简单且功能强大。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“METRIC仪表盘模板>日志生成指标任务监控中心”仪表盘，查看图表详情。

----结束

日志生成指标任务监控中心仪表盘中的过滤器说明如下所示：

规则ID图展示不同的规则ID，所关联的查询分析语句如下所示：

```
select distinct(task_set)
```

日志生成指标任务监控中心仪表盘中的重要图表说明如下所示：

- **输入行数**图展示输入行数的变化情况，所关联的查询分析语句如下所示：

```
SELECT CASE   WHEN  "input" < 1000 THEN concat( cast( "input" AS VARCHAR ), '行' )  WHEN  "input" < 1000 * 1000 THEN concat( cast( round( "input"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN
```



```
"input" < 1000000000 THEN concat( cast( round( "input"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "input"/ 1000.0 < 1000000000 THEN concat( cast( round( "input"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "input"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("input") as "input")
```

- **输出行数**图展示输出行数的变化情况，所关联的查询分析语句如下所示

```
SELECT CASE WHEN "output" < 1000 THEN concat( cast( "output" AS VARCHAR ), '行' ) WHEN "output" < 1000 * 1000 THEN concat( cast( round( "output"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "output" < 1000000000 THEN concat( cast( round( "output"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "output"/ 1000.0 < 1000000000 THEN concat( cast( round( "output"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "output"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("output") as "output")
```

- **满足过滤条件行数**图展示满足过滤条件行数的变化情况，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "filters" < 1000 THEN concat( cast( "filters" AS VARCHAR ), '行' ) WHEN "filters" < 1000 * 1000 THEN concat( cast( round( "filters"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "filters" < 1000000000 THEN concat( cast( round( "filters"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filters"/ 1000.0 < 1000000000 THEN concat( cast( round( "filters"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filters"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("filters") as "filters")
```

- **不满足过滤条件行数**图展示不满足过滤条件行数的变化情况，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "filter_drops" < 1000 THEN concat( cast( "filter_drops" AS VARCHAR ), '行' ) WHEN "filter_drops" < 1000 * 1000 THEN concat( cast( round( "filter_drops"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "filter_drops" < 1000000000 THEN concat( cast( round( "filter_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "filter_drops"/ 1000.0 < 1000000000 THEN concat( cast( round( "filter_drops"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "filter_drops"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("filter_drops") as "filter_drops")
```

- **采样行数**图展示采样行数的分布情况，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "samples" < 1000 THEN concat( cast( "samples" AS VARCHAR ), '行' ) WHEN "samples" < 1000 * 1000 THEN concat( cast( round( "samples"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "samples" < 1000000000 THEN concat( cast( round( "samples"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "samples"/ 1000.0 < 1000000000 THEN concat( cast( round( "samples"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "samples"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("samples") as "samples")
```

- **采样丢弃行数**图展示采样丢弃行数的分布情况，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "sample_drops" < 1000 THEN concat( cast( "sample_drops" AS VARCHAR ), '行' ) WHEN "sample_drops" < 1000 * 1000 THEN concat( cast( round( "sample_drops"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "sample_drops" < 1000000000 THEN concat( cast( round( "sample_drops"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "sample_drops"/ 1000.0 < 1000000000 THEN concat( cast( round( "sample_drops"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "sample_drops"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum( "sample_drops" ) as "sample_drops")
```

- **超过日志时间范围行数**图展示超过日志时间范围行数的分布情况，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN "out_of_bounds" < 1000 THEN concat( cast( "out_of_bounds" AS VARCHAR ), '行' ) WHEN "out_of_bounds" < 1000 * 1000 THEN concat( cast( round( "out_of_bounds"/ 1000, 1 ) AS VARCHAR ), '千行' ) WHEN "out_of_bounds" < 1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000000.0, 1 ) AS VARCHAR ), '百万行' ) WHEN "out_of_bounds"/ 1000.0 < 1000000000 THEN concat( cast( round( "out_of_bounds"/ 1000 / 1000000.0, 1 ) AS VARCHAR ), '十亿行' ) ELSE concat( cast( round( "out_of_bounds"/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), '万亿行' ) END AS "total" from (select sum("out_of_bounds") as "out_of_bounds")
```

- **执行记录**图展示执行记录的分布情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT( "__time", 'yyyy-MM-dd HH:mm:ss:SSS', '+08:00') as "统计时间", sum("input") as "输入行数",sum("output") as "输出行数",sum("filters") as "满足过滤条件行数",sum("filter_drops") as "不满足过滤条件行数",sum("samples") as "采样行数",sum("sample_drops") as "采样丢弃行数",sum("out_of_bounds") as "超过日志时间范围行数" group by __time order by __time desc limit 1000
```


7.3.3.15 NGINX 仪表盘模板

日志服务支持采集NGINX日志，并进行多维度分析。NGINX仪表盘模板分组里有三种仪表盘模板，分别是NGINX监控中心、NGINX访问中心和NGINX秒级监控。

7.3.3.15.1 NGINX 秒级监控

日志服务支持采集NGINX日志，并进行多维度分析。云日志服务支持日志采集向导一站式采集NGINX日志，并为NGINX日志配置结构化和仪表盘。该仪表盘主要展示Upstream状态码等信息，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

日志配置结构化，详情请参见**结构化配置**。

背景信息

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板>NGINX秒级监控”仪表盘，查看图表详情。

----结束

NGINX秒级监控仪表盘中的重要图表说明如下所示：

- QPS图展示QPS的变化情况，所关联的查询分析语句如下所示：
```
SELECT TIME_FORMAT(TIME_CEIL(__time,'PT1S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_ , COUNT(*) as QPS from log group by _time_
```
- 成功率图展示成功率的变化情况，所关联的查询分析语句如下所示：
```
select __time,round(CASE WHEN "成功率" > 0 THEN "成功率" else 0 end,2) as "成功率" from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1) as '成功率' from log group by __time)
```
- 延迟图展示访问延时的变化情况，所关联的查询分析语句如下所示：
```
select __time,round(CASE WHEN "访问延迟" > 0 THEN "访问延迟" else 0 end,2) as "访问延迟",round(CASE WHEN "Upstream延迟" > 0 THEN "Upstream延迟" else 0 end,2) as "Upstream延迟" from (select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time, avg(request_time)* 1000 as '访问延迟',avg(upstream_response_time)* 1000 as 'Upstream延迟' from log group by __time)
```
- 流量图展示请求流量和返回body流量的变化情况，所关联的查询分析语句如下所示：
```
select TIME_FORMAT(TIME_CEIL(__time,'PT5S'),'yyyy-MM-dd HH:mm:ss','+08:00') as __time , sum("request_length") as "请求流量", sum("body_bytes_sent") as "返回body流量" group by __time
```
- 状态码图展示响应状态码的变化情况，所关联的查询分析语句如下所示：
```
select t.t as "time",
    CASE WHEN a."2XX" IS NOT NULL THEN  CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",
    CASE WHEN b."3XX" IS NOT NULL THEN  CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",
    CASE WHEN c."4XX" IS NOT NULL THEN  CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",
    CASE WHEN d."5XX" IS NOT NULL THEN  CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",
    CASE WHEN e."其他" IS NOT NULL THEN  CAST(e."其他" AS BIGINT)  ELSE 0 END as "其他"
    from (select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc  ) t left join (select
```



```
TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "status" >= 200 and "status" < 300 group by t order by t asc ) a on t.t =a.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "status" >= 300 and "status" < 400 group by t order by t asc) b on t.t =b.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "status" >= 400 and "status" < 500 group by t order by t asc) c on t.t =c.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "status" >= 500 and "status" < 600 group by t order by t asc) d on t.t =d.t left join (select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "status" < 200 or "status" >= 600 group by t order by t asc) e on t.t =e.t
```

- 后端响应码图展示后端响应状态码的变化情况，所关联的查询分析语句如下所示：

```
select t.t as "time",
    CASE WHEN a."2XX" IS NOT NULL THEN  CAST(a."2XX" AS BIGINT) ELSE 0 END as "2XX",
    CASE WHEN b."3XX" IS NOT NULL THEN  CAST(b."3XX" AS BIGINT) ELSE 0 END as "3XX",
    CASE WHEN c."4XX" IS NOT NULL THEN  CAST(c."4XX" AS BIGINT) ELSE 0 END as "4XX",
    CASE WHEN d."5XX" IS NOT NULL THEN  CAST(d."5XX" AS BIGINT) ELSE 0 END as "5XX",
    CASE WHEN e."其他" IS NOT NULL THEN  CAST(e."其他" AS BIGINT)  ELSE 0 END as "其他"
    from (
    select TIME_CEIL(__time,'PT5S') as t from log group by t order by t asc
    ) t
    left join(
    select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "2XX" from log WHERE "upstream_status" >= 200 and "upstream_status" < 300 group by t order by t asc) a
    on t.t = a.t
    left join (
    select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "3XX" from log WHERE "upstream_status" >= 300 and "upstream_status" < 400 group by t order by t asc) b
    on t.t =b.t
    left join (
    select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "4XX" from log WHERE "upstream_status" >= 400 and "upstream_status" < 500 group by t order by t asc) c
    on t.t =c.t
    left join (
    select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "5XX" from log WHERE "upstream_status" >= 500 and "upstream_status" < 600 group by t order by t asc) d
    on t.t =d.t
    left join (
    select TIME_CEIL(__time,'PT5S') as t , CAST(COUNT(1) as VARCHAR) as "其他" from log WHERE "upstream_status" < 200 or "upstream_status" >= 600 group by t order by t asc) e
    on t.t =e.t
```

7.3.3.15.2 NGINX 访问中心

云日志服务支持日志采集向导一站式采集NGINX日志，并为NGINX日志配置结构化和仪表盘。该仪表盘主要展示NGINX日志的PV对比、访问量PV分布（中国）、访问量PV分布（世界）、访问量UV分布（中国）、等信息，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

日志配置结构化，详情请参见**结构化配置**。

背景信息

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板>NGINX访问中心”仪表盘，查看图表详情。

----结束

NGINX访问中心仪表盘中的重要图表说明如下所示：

- PV对比昨日图展示PV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 86400) as diff from (select count(1) as "pv" from log))
  ```
- PV对比上周图展示PV数对比上周的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "pv" , 604800) as diff from (select count(1) as "pv" from log))
  ```
- UV对比昨日图展示UV数对比昨日的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 86400) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
  ```
- UV对比上周图展示UV数对比上周的变化情况，所关联的查询分析语句如下所示：
  ```
  select diff[1] as "total", round((diff[1] - diff[2]) / diff[2] * 100, 2) as inc from(select compare( "uv" , 604800) as diff from (select APPROX_COUNT_DISTINCT(my_remote_addr) as "uv" from log))
  ```
- 访问量PV分布(中国)图展示中国区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
  ```
  select ip_to_province(remote_addr) as province, count(1) as pv where IP_TO_COUNTRY (remote_addr) = '中国'  group by province HAVING province not in ('','保留地址','*')
  ```
- 访问量PV分布(世界)图展示世界区域内访问量PV的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT ip_to_country(remote_addr) as country,COUNT(1) as PV GROUP BY country HAVING country not in ('','保留地址','*')
  ```
- 访问量UV分布(中国)图展示中国区域内访问量UV的分布情况，所关联的查询分析语句如下所示：
  ```
  select ip_to_province(remote_addr) as province, APPROX_COUNT_DISTINCT(remote_addr) as UV where IP_TO_COUNTRY (remote_addr) = '中国'  group by province HAVING province not in ('','保留地址','*')
  ```
- 访问量UV分布(世界)图展示世界区域内访问量UV的分布情况，所关联的查询分析语句如下所示：
  ```
  select ip_to_country(remote_addr) as country, APPROX_COUNT_DISTINCT(remote_addr) as uv group by country HAVING country not in ('','保留地址','*')
  ```
- 平均时延分布（中国）图展示中国区域内平均时延的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT province,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)"FROM (SELECT ip_to_province(remote_addr) as province,avg(request_time) * 1000 AS "平均延迟(ms)"WHERE  IP_TO_COUNTRY (remote_addr) = '中国'GROUP BY province HAVING province not in ('','保留地址','*'))
  ```
- 平均时延分布（世界）图展示世界区域内平均时延的分布情况，所关联的查询分析语句如下所示：
  ```
  SELECT country,round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 2 ) AS "平均延迟(ms)"FROM (SELECT ip_to_country(remote_addr) as country,avg(request_time) * 1000 AS "平均延迟(ms)" GROUP BY country HAVING country not in ('','保留地址','*'))
  ```
- 今日PV/UV图展示最近一天的PV/UV数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(__time,'PT600S') AS _time_ , count(1) as PV, APPROX_COUNT_DISTINCT(my_remote_addr) as UV from log WHERE __time <= CURRENT_TIMESTAMP  and __time >= DATE_TRUNC( 'DAY',(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR group by _time_ order by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
  ```
- 7日PV/UV图展示七天的PV/UV数，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV,UV FROM (select TIME_CEIL(__time,'PT600S') AS _time_ , count(1) as PV,  APPROX_COUNT_DISTINCT(remote_addr) as UV from log WHERE __time <= CURRENT_TIMESTAMP  and __time >= DATE_TRUNC( 'DAY',
  ```



```
(CURRENT_TIMESTAMP + INTERVAL '8' HOUR)) - INTERVAL '8' HOUR - INTERVAL '7' DAY group by _time_ order by _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- 区域访问TOP10(省份)图展示访问排名前十的省份，所关联的查询分析语句如下所示：
```
select ip_to_province(remote_addr) as "province", count(1) as "访问次数" group by "province" HAVING "province" <> '-1' order by "访问次数" asc limit 10
```
- 区域访问TOP10(城市)图展示访问排名前十的城市，所关联的查询分析语句如下所示：
```
select ip_to_city(remote_addr) as "city", count(1) as "访问次数" group by "city" HAVING  "city" <> '-1' order by "访问次数" asc  limit 10
```
- Host访问TOP10图展示访问排名前十的主机，所关联的查询分析语句如下所示：
```
select  host as "Host", count(1) as "PV" group by "Host" order by "PV" asc limit 10
```
- UserAgent访问TOP10图展示访问排名前十的UserAgent，所关联的查询分析语句如下所示：
```
select http_user_agent as "UserAgent", count(1) as "PV" group by "UserAgent" order by "PV" asc limit 10
```
- 设备占比(终端)图展示各终端设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|android|ios') then '移动端' else 'PC端' end as type , count(1) as total group by  type
```
- 设备占比(系统)图展示各系统设备的访问占比，所关联的查询分析语句如下所示：
```
select case when regexp_like(lower(http_user_agent), 'iphone|ipod|ios') then 'IOS' when regexp_like(lower(http_user_agent), 'android') then 'Android' else 'other' end as type , count(1) as total group by  type HAVING type != 'other'
```
- TOP URL图展示访问前十url的PV、UV及访问成功率等信息，所关联的查询分析语句如下所示：
```
select request_uri , count(1) as PV, APPROX_COUNT_DISTINCT(remote_addr) as UV, round(sum( case when status < 400 then 1 else 0 end ) * 100.0 / count(1), 2) as "访问成功率" group by request_uri ORDER by PV desc
```
- TOP 访问IP图展示访问前十的IP及城市、运营商和PV等数据，所关联的查询分析语句如下所示：
```
select remote_addr as "来源IP",ip_to_country(remote_addr) as "国家",ip_to_province(remote_addr) as "省份",ip_to_city(remote_addr) as "城市",ip_to_provider(remote_addr) as "运营商",count(1) as "PV",http_user_agent as "UserAgent采样",request_uri as "URL采样" group by remote_addr,http_user_agent,request_uri ORDER by "PV" desc
```

7.3.3.15.3 NGINX 监控中心

云日志服务支持日志采集向导一站式采集ELB日志，并为NGINX日志配置结构化和仪表盘。该仪表盘主要展示NGINX日志的访问量PV、访问量UV、流量、访问失败率、延迟等指标，全方位展示网站访问情况。您还可以使用云日志服务的查询分析语句，分析网站的延时情况，及时调优网站。

前提条件

日志配置结构化，详情请参见**结构化配置**。

背景信息

Nginx (engine x) 是一个高性能的HTTP和反向代理web服务器，同时也提供了IMAP/POP3/SMTP服务。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“NGINX仪表盘模板>NGINX监控中心”仪表盘，查看图表详情。

----结束

NGINX监控中心仪表盘中重要图表说明如下所示：

- 访问量PV图展示访问量的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,PV FROM ( SELECT TIME_CEIL ( __time, 'PT300S' ) AS _time_, count( 1 ) AS PV FROM log GROUP BY _time_ ) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100 OFFSET 1
  ```
- 请求成功率图展示请求成功率的变化情况，所关联的查询分析语句如下所示：
  ```
  select ROUND(sum(case when status < 400 then 1 else 0 end) * 100.0 / count(1),2) as cnt
  ```
- 平均延迟图展示平均延迟的变化情况，所关联的查询分析语句如下所示：
  ```
  select round(avg(request_time) * 1000, 3) as cnt
  ```
- 4XX请求数图展示4xx的请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" >= 400 and "status" < 500
  ```
- 404请求数图展示404请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 404
  ```
- 429请求数图展示429请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 429
  ```
- 504请求数图展示504请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT COUNT(1) as cnt WHERE "status" = 504
  ```
- 5XX请求数图展示5xx的请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT(TIME_CEIL(__time,'PT300S'),'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_ , count(1) as cnt where "status" >= 500 group by _time_
  ```
- 状态码分布图展示请求状态码的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT status, COUNT(1) AS rm GROUP BY status
  ```
- 访问量UV图展示访问量UV的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,UV FROM (select TIME_CEIL(__time,'PT600S') AS _time_ , APPROX_COUNT_DISTINCT(remote_addr) as UV  from log group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
  ```
- 流量图展示入流量和出流量的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT(_time_,'yyyy-MM-dd HH:mm:ss','+08:00') AS _time_,round( CASE WHEN "入流量" > 0 THEN "入流量" ELSE 0 END, 2 ) AS "入流量",round( CASE WHEN "出流量" > 0 THEN "出流量" ELSE 0 END, 2 ) AS "出流量" FROM (SELECT TIME_CEIL(__time,'PT600S') AS _time_,sum(request_length) / 1024.0 AS "入流量",sum(bytes_sent) / 1024.0 AS "出流量" group by _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
  ```
- 访问失败率图展示访问失败率和5xx的变化情况，所关联的查询分析语句如下所示：
  ```
  SELECT TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "失败率" > 0 THEN "失败率" ELSE 0 END, 2 ) AS "失败率",round( CASE WHEN "5XX比例" > 0 THEN "5XX比例" ELSE 0 END, 2 ) AS "5XX比例" from (select TIME_CEIL(__time,'PT600S') AS _time_,sum(case when status >= 400 then 1 else 0 end) * 100.0 / count(1) as '失败率' , sum(case when status >=500 THEN 1 ELSE 0 END)*100.0/COUNT(1) as '5XX比例' group by  _time_) WHERE _time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
  ```
- 延迟图展示访问P50、P90、P99、P9999延迟的变化情况，所关联的查询分析语句如下所示：
  ```
  select TIME_FORMAT( _time_, 'yyyy-MM-dd HH:mm:ss', '+08:00' ) as _time_,round( CASE WHEN "平均" > 0 THEN "平均" ELSE 0 END, 2 ) AS "平均",round( CASE WHEN "P50" > 0 THEN "P50" ELSE 0 END, 2 ) AS "P50",round( CASE WHEN "P90" > 0 THEN "P90" ELSE 0 END, 2 ) AS "P90",round( CASE WHEN "P99" > 0 THEN "P99" ELSE 0 END, 2 ) AS "P99",round( CASE WHEN "P9999" > 0 THEN "P9999" ELSE 0 END, 2 ) AS "P9999" from (select TIME_CEIL(__time,'PT600S') as _time_,avg(request_time) * 1000 as "平均", APPROX_QUANTILE_DS("request_time", 0.50)*1000 as
  ```



```
"P50", APPROX_QUANTILE_DS("request_time", 0.90)*1000 as
"P90" ,APPROX_QUANTILE_DS("request_time", 0.99)*1000 as
'P99',APPROX_QUANTILE_DS("request_time", 0.9999)*1000 as 'P9999' group by  _time_) WHERE
_time_ <= CURRENT_TIMESTAMP LIMIT 100000 OFFSET 1
```

- Host请求TOP图展示主机请求TOP信息的变化情况，所关联的查询分析语句如下所示：

```
SELECT "host", pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,
2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS
"入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量
(KB)"  FROM ( SELECT "host", count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr ) AS uv,
sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",
avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",
sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY "host" ) ORDER BY pv
DESC
```

- Host延迟TOP图展示主机延迟TOP信息的变化情况，所关联的查询分析语句如下所示：

```
SELECT "host", pv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 )
AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99
延迟(ms)" FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0
END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟
(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY
"host" ) ORDER BY "平均延迟(ms)" desc
```

- Host失败率TOP图展示主机访问失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT "host", pv,round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 )
AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 )
AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS
"P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99
延迟(ms)"  FROM ( SELECT "host", count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE
0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY
"host"  ) ORDER BY "访问失败率(%)" desc
```

- URL请求TOP图展示URL请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT request_uri, pv,uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0
END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0
END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END,
3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出
流量(KB)"  FROM ( SELECT request_uri, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr )
AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率
(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)",
sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY request_uri  ) ORDER BY
pv desc
```

- URL延迟TOP图展示URL延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT request_uri, pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END,
2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )
AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS
"P99延迟(ms)" FROM (SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" < 400 THEN
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟
(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",
APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY
request_uri  ) ORDER BY "平均延迟(ms)" desc
```

- URL失败率TOP图展示URL失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT request_uri, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END,
2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END,
3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 )
AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS
"P99延迟(ms)" FROM( SELECT request_uri, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN
```



```
1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' GROUP BY request_uri  )ORDER BY "访问失败率(%)" desc
```

- 后端请求TOP图展示后端请求TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, uv, round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "入流量(KB)" > 0 THEN "入流量(KB)" ELSE 0 END, 3 ) AS "入流量(KB)", round( CASE WHEN "出流量(KB)" > 0 THEN "出流量(KB)" ELSE 0 END, 3 ) AS "出流量(KB)"  FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, APPROX_COUNT_DISTINCT ( remote_addr ) AS uv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", sum( request_length ) / 1024.0 AS "入流量(KB)", sum( bytes_sent ) / 1024.0 AS "出流量(KB)"  WHERE "host" != ''  GROUP BY addr having length(upstream_addr) > 2) ORDER BY "pv" desc
```

- 后端延迟TOP图展示后端延迟TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr,pv,round( CASE WHEN "访问成功率(%)" > 0 THEN "访问成功率(%)" ELSE 0 END, 2 ) AS "访问成功率(%)",round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)",round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)",round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)" FROM (SELECT upstream_addr as addr,count( 1 ) AS pv,sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问成功率(%)",avg( request_time ) * 1000 AS "平均延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)",APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != '' GROUP BY addr having length(upstream_addr) > 2) ORDER BY "平均延迟(ms)" desc
```

- 后端失败率TOP图展示后端失败率TOP的变化情况，所关联的查询分析语句如下所示：

```
SELECT addr, pv, round( CASE WHEN "访问失败率(%)" > 0 THEN "访问失败率(%)" ELSE 0 END, 2 ) AS "访问失败率(%)", round( CASE WHEN "平均延迟(ms)" > 0 THEN "平均延迟(ms)" ELSE 0 END, 3 ) AS "平均延迟(ms)", round( CASE WHEN "P90延迟(ms)" > 0 THEN "P90延迟(ms)" ELSE 0 END, 3 ) AS "P90延迟(ms)", round( CASE WHEN "P99延迟(ms)" > 0 THEN "P99延迟(ms)" ELSE 0 END, 3 ) AS "P99延迟(ms)"  FROM ( SELECT upstream_addr as addr, count( 1 ) AS pv, sum( CASE WHEN "status" >= 400 THEN 1 ELSE 0 END ) * 100.0 / count( 1 ) AS "访问失败率(%)", avg( request_time ) * 1000 AS "平均延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.9) * 1000 AS "P90延迟(ms)", APPROX_QUANTILE_DS(request_time, 0.99) * 1000 AS "P99延迟(ms)" WHERE "host" != ''  GROUP BY addr having length(upstream_addr) > 2)ORDER BY "访问失败率(%)" desc
```

7.3.3.16 VPC 仪表盘模板

云日志服务支持日志采集向导一站式采集VPC日志，并为VPC日志配置结构化和仪表盘。该仪表盘主要展示VPC日志的Action总次数，ACCEPT总字节数、ACCEPT总包数、REJECT总字节数、REJECT总包数、源地址的Action次数分布、总分钟Action次数、Action分布、流日志记录状态分布、Action次数的源地址运行商分布、Top5字节数的源地址、Top5字节数的目标地址、Top5包数的目标端口、各协议的每分钟包数、弹性网卡。

前提条件

- 已采集VPC日志，详情请参见**虚拟私有云VPC接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

虚拟私有云（Virtual Private Cloud）是用户在华为云上申请的隔离的、私密的虚拟网络环境。用户可以自由配置VPC内的IP地址段、子网、安全组等子服务，也可以申请弹性带宽和弹性公网IP搭建业务系统。 VPC日志流中记录了虚拟私有云中的流量信息，可以帮助您检查和优化安全组和网络ACL控制规则、监控网络流量、进行网络攻击分析等。

分析网站访问情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志管理”。

步骤2 在“日志应用”模块中，单击“VPC日志流中心”，选择“进入仪表盘”。

步骤3 在仪表盘模板下方，选择“VPC仪表盘模板>VPC流日志”仪表盘，查看图表详情。

----结束

VPC流日志仪表盘中的重要图表说明如下所示：

- Action总次数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时候的总数量，所关联的查询分析语句如下所示：
```
select CASE WHEN total_actions < 1000 THEN concat(cast( total_actions AS VARCHAR), '次') WHEN total_actions < 1000 * 1000 THEN concat(cast(round(total_actions / 1000.0, 2) AS VARCHAR),'千次') WHEN total_actions < 1000000000 THEN concat(cast(round(total_actions / 1000000.0, 2) AS VARCHAR),'百万次') WHEN total_actions / 1000.0 < 1000000000 THEN concat(cast(round(total_actions / 1000 / 1000000.0, 1) AS VARCHAR),'十亿次') ELSE concat(cast(round(total_actions / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR),'万亿次') END AS "total_actions" from (select count(1) as total_actions where log_status='OK' and version=1)
```
- ACCEPT总字节数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1并且是安全组或网络ACL允许记录的流量时数据包的总大小，所关联的查询分析语句如下所示：
```
select CASE WHEN accept_bytes < 1024 THEN concat(cast( accept_bytes AS VARCHAR), 'B') WHEN accept_bytes < 1024 * 1024 THEN concat(cast(round(accept_bytes / 1024, 2) AS VARCHAR),'KB') WHEN accept_bytes < 1000000000 THEN concat(cast(round(accept_bytes /1024.0 /1024, 2) AS VARCHAR),'MB') WHEN accept_bytes / 1000.0 < 1000000000 THEN concat(cast(round(accept_bytes / 1024 / 1000000.0, 2) AS VARCHAR),'GB') ELSE concat(cast(round(accept_bytes / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR),'TB') END AS "accept_bytes" from (select sum(bytes) as accept_bytes where log_status='OK' and version=1 and action='ACCEPT')
```
- ACCEPT总包数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1并且是安全组或网络ACL允许记录的流量时数据包的总数量，所关联的查询分析语句如下所示：
```
select CASE WHEN accept_packets < 1024 THEN concat(cast( accept_packets AS VARCHAR), 'B') WHEN accept_packets < 1024 * 1024 THEN concat(cast(round(accept_packets / 1024, 2) AS VARCHAR),'KB') WHEN accept_packets < 1000000000 THEN concat(cast(round(accept_packets / 1024.0 /1024, 2) AS VARCHAR),'MB') WHEN accept_packets / 1000.0 < 1000000000 THEN concat(cast(round(accept_packets / 1024 / 1000000.0, 2) AS VARCHAR),'GB') ELSE concat(cast(round(accept_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR),'TB') END AS "accept_packets" from (select sum(packets) as accept_packets where log_status='OK' and version=1 and action='ACCEPT')
```
- REJECT总字节数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1并且是安全组或网络ACL拒绝记录的流量时数据包的总大小，所关联的查询分析语句如下所示：
```
select CASE WHEN reject_bytes < 1024 THEN concat(cast( reject_bytes AS VARCHAR), 'B') WHEN reject_bytes < 1024 * 1024 THEN concat(cast(round(reject_bytes / 1024, 2) AS VARCHAR),'KB') WHEN reject_bytes < 1000000000 THEN concat(cast(round(reject_bytes /1024.0 /1024, 2) AS VARCHAR),'MB') WHEN reject_bytes / 1000.0 < 1000000000 THEN concat(cast(round(reject_bytes / 1024 / 1000000.0, 2) AS VARCHAR),'GB') ELSE concat(cast(round(reject_bytes / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR),'TB') END AS "reject_bytes" from (select sum(bytes) as reject_bytes where log_status='OK' and version=1 and action='REJECT')
```
- REJECT总包数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1并且是安全组或网络ACL拒绝记录的流量时数据包的总数量，所关联的查询分析语句如下所示：
```
select CASE WHEN reject_packets < 1024 THEN concat(cast( reject_packets AS VARCHAR), 'B') WHEN reject_packets < 1024 * 1024 THEN concat(cast(round(reject_packets / 1024, 2) AS VARCHAR),'KB') WHEN reject_packets < 1000000000 THEN concat(cast(round(reject_packets /1024.0 /1024, 2) AS VARCHAR),'MB') WHEN reject_packets / 1000.0 < 1000000000 THEN concat(cast(round(reject_packets / 1024 / 1000000.0, 2) AS VARCHAR),'GB') ELSE concat(cast(round(reject_packets / 1000.0 / 1000 / 1000 / 1000, 1) AS VARCHAR),'TB') END AS "reject_packets" from (select sum(packets) as reject_packets where log_status='OK' and version=1 and action='REJECT')
```


- 源地址的Action次数分布图展示中国区域内不同源地址的访问次数，所关联的查询分析语句如下所示：
```
select IP_TO_PROVINCE(srcaddr) as province, count(1) as total_actions where IP_TO_COUNTRY (srcaddr) = '中国'  group by province HAVING province not in ('','保留地址','*')
```
- 每分钟Action次数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同流量关联的操作的个数，所关联的查询分析语句如下所示：
```
select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)),'MM-dd HH:mm') as "t", "action", count(1) as "total_actions" where log_status='OK' and version=1 group by "t", "action" order by t asc limit 1000
```
- Action分布图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同与流量关联操作的分布情况，所关联的查询分析语句如下所示：
```
select action, count(1) as total_actions where log_status='OK' and version=1 group by action
```
- 流日志记录状态分布图展示VPC流日志版本为1时流日志的日志状态的分布情况，所关联的查询分析语句如下所示：
```
select log_status, count(1) as total_actions where version=1 group by log_status
```
- Action次数的源地址运营商分布图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同源地址运营商的分布情况，所关联的查询分析语句如下所示：
```
select ip_to_provider(srcaddr) as src_addr_provider, count(1) as total_actions where log_status='OK' and version=1 group by src_addr_provider order by total_actions desc limit 5
```
- Top5字节数的源地址图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同源地址字节数Top5，所关联的查询分析语句如下所示：
```
select ip_to_provider(srcaddr) as src_addr_provider, count(1) as total_actions where log_status='OK' and version=1 group by src_addr_provider order by total_actions desc limit 5
```
- Top5字节数的目标地址图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同目标地址字节数Top5，所关联的查询分析语句如下所示：
```
select dstaddr, sum(bytes) as total_bytes where log_status='OK' and version=1 group by dstaddr order by total_bytes desc limit 5
```
- Top5包数的目标端口图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同目标端口的包数Top5，所关联的查询分析语句如下所示：
```
select dstport, sum(packets) as total_packets where log_status='OK' and version=1 group by dstport order by total_packets desc limit 5
```
- 各协议的每分钟包数图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同IANA协议编号在每分钟的包数，所关联的查询分析语句如下所示：
```
select TIME_FORMAT(date_trunc('minute', MILLIS_TO_TIMESTAMP("start" * 1000)),'MM-dd HH:mm') as t, protocol, sum(packets) as total_packets where log_status='OK' and version=1 group by t, protocol order by t asc limit 1000
```
- 弹性网卡图展示当流日志的日志状态为数据正常记录到选定目标并且VPC流日志版本为1时不同记录流量的网卡的ID的数据包总数量和数据包总大小，所关联的查询分析语句如下所示：
```
select interface_id as "ID", sum(packets) as '数据包总数量', sum(bytes) as '数据包总大小' where log_status='OK' and version=1 group by "ID"
```

7.3.3.17 WAF 仪表盘模板

7.3.3.17.1 WAF 安全日志中心

WAF安全日志中心仪表盘主要展示攻击网站、攻击来源地区、攻击类型、攻击拦截情况等。

前提条件

- 已采集WAF日志，详情请参考**Web应用防火墙WAF接入LTS**。
- 日志配置结构化，详情请参见**结构化配置**。

背景信息

Web应用防火墙（Web Application Firewall， WAF），通过对HTTP(S)请求进行检测，识别并阻断SQL注入、跨站脚本攻击、网页木马上传、命令/代码注入、文件包含、敏感文件访问、第三方应用漏洞攻击、CC攻击、恶意爬虫扫描、跨站请求伪造等攻击，保护Web服务安全稳定。

分析网站被攻击情况

步骤1 登录云日志服务控制台，在左侧导航栏中选择“仪表盘 ”。

步骤2 在仪表盘模板下方，选择“WAF仪表盘模板>WAF安全日志中心”仪表盘，查看图表详情。

----结束

WAF安全日志中心仪表盘中的重要图表说明如下所示：

- 被攻击网站图展示被攻击的网站数量以及前一天同时段的趋势对比，所关联的查询分析语句如下所示：

```
SELECT diff [ 1 ] AS  "VALUE", COALESCE ( diff [ 1 ]- diff [ 2 ], 0 ) AS "BEFORE" FROM
    (
    SELECT
    compare ( "DATA", 86400 ) AS diff
    FROM
    ( SELECT count( DISTINCT "host" ) AS "DATA" FROM log
    WHERE action != ''
    ))
```

- 攻击来源国家图展示攻击来源的IP隶属不同国家的数量，所关联的查询分析语句如下所示：

```
SELECT
    diff [ 1 ] AS
    "VALUE"
    ,
    COALESCE ( diff [ 1 ]- diff [ 2 ], 0 ) AS "BEFORE"
    FROM
    (
    SELECT
    compare ( "DATA", 86400 ) AS diff
    FROM
    ( SELECT count( DISTINCT ip_to_country ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END) )
AS "DATA" FROM log
    WHERE action != ''
    )
    )
```

- Web攻击拦截图展示所选时间段Web攻击拦截次数以及前一天同时段的趋势对比，所关联的查询分析语句如下所示：

```
SELECT
    CASE

    WHEN
    diff [ 1 ] < 1000 THEN
    concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
    WHEN diff [ 1 ] < 1000 * 1000 THEN
    concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
    WHEN diff [ 1 ] < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
    WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
```



```
    concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
    END AS
    "value"
    ,
    CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
    FROM
    ( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log
WHERE action = '' ) )
```

- **CC攻击拦截**图展示时间段内攻击次数以及与前一天同时段的数据变化，所关联的查询分析语句如下所示：

```
SELECT
    CASE

    WHEN
    diff [ 1 ] < 1000 THEN
    concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
    WHEN diff [ 1 ] < 1000 * 1000 THEN
    concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
    WHEN diff [ 1 ] < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
    WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
    END AS
    "value"
    ,
    CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
    FROM
    ( SELECT compare ( "data", 86400 ) AS diff FROM ( SELECT count( 1 ) AS "data" FROM log
WHERE attack != 'default') )
```

- **攻击者UV**图展示攻击者数量情况，所关联的查询分析语句如下所示：

```
SELECT
    CASE
    WHEN
    diff [ 1 ] < 1000 THEN
    concat( cast( cast ( diff [ 1 ] AS INTEGER ) AS VARCHAR ), ' 个' )
    WHEN diff [ 1 ] < 1000 * 1000 THEN
    concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千个' )
    WHEN diff [ 1 ] < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万个' )
    WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿' ) ELSE
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿' )
    END AS "value",
    CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio
    FROM
    (
    SELECT
    compare ( "data", 86400 ) AS diff
    FROM
    ( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS "data"
    FROM log
    ))
```

- 攻击拦截图展示所选时间段内的攻击拦截次数以及与前一天同时段数据的趋势对比，所关联的查询分析语句如下所示：

```
SELECT
    CASE
    WHEN
    diff [ 1 ] < 1000 THEN
    concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' )
    WHEN diff [ 1 ] < 1000 * 1000 THEN
    concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' )
    WHEN diff [ 1 ] < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' )
    WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN
    concat( cast( round( diff [ 1 ]/ 1000.0 / 1000000, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE
```



```
concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' )
    END AS
    "value",
    CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS "ratio"
    FROM
    (
    SELECT
    compare ( "data", 86400 ) AS diff
    FROM
    ( SELECT count( 1 ) AS "data" FROM log WHERE action != '')
    )
```

- CC攻击展示攻击者的国内ip分布，所关联的查询分析语如下所示：

```
SELECT
    ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
    count( 1 ) AS "攻击次数"
    WHERE attack != 'default' and ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END)
= '中国'
    GROUP BY
    province
```

- **攻击类型分布**图展示不同攻击类型随时间的攻击次数分布，所关联的查询分析语句如下所示：

```
SELECT time_format( MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600)), 'HH:mm' ) AS dt, count( 1 ) AS cnt, CASE WHEN
action = 'block' THEN '拦截'  WHEN action = 'log' THEN '仅记录'  WHEN action = 'captcha' THEN '人机
验证' END AS attack FROM log WHERE action != '' GROUP BY TIMESTAMP_TO_MILLIS(__time) -
MOD(TIMESTAMP_TO_MILLIS(__time), 3600), attack ORDER BY cnt DESC
```

- **Web攻击**图展示Web攻击的来源IP在国内的地域分布情况，所关联的查询分析语句如下所示：

```
SELECT
    ip_to_province (
    CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS province,
    count( 1 ) AS "攻击次数"
    WHERE action = 'block' and ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END) =
'中国'
    GROUP BY
    province
```

- CC攻击(世界)图展示CC攻击的来源IP在全世界的地域分布情况，所关联的查询分析语句如下所示：

```
SELECT
    ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
    count( 1 ) AS "攻击次数"
    WHERE attack != 'default'
    GROUP BY
    country
```

- Web攻击(世界)图展示Web攻击的来源IP在全世界的地域分布情况，所关联的查询分析语句如下所示：

```
SELECT
    ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
    count( 1 ) AS "攻击次数"
    WHERE action = 'block'
    GROUP BY
    country
```

7.3.3.17.2 WAF 访问日志中心

WAF访问日志中心仪表盘主要展示PV、UV、流量带宽趋势、PV/UV趋势、访问状态分布、访问来源等。

前提条件

- 已采集WAF日志，详情请参考**Web应用防火墙WAF接入LTS**。

- 日志配置结构化，详情请参见**结构化配置**。

背景信息

Web应用防火墙（Web Application Firewall， WAF），通过对HTTP(S)请求进行检测，识别并阻断SQL注入、跨站脚本攻击、网页木马上传、命令/代码注入、文件包含、敏感文件访问、第三方应用漏洞攻击、CC攻击、恶意爬虫扫描、跨站请求伪造等攻击，保护Web服务安全稳定。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“WAF仪表盘模板>WAF访问日志中心”仪表盘，查看图表详情。

----结束

WAF访问日志中心仪表盘中的重要图表说明如下所示：

- PV图表展示所选时间内的总调用次数以及与前一天同时段的趋势对比，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] < 1000000000 THEN concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' ) WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN concat( cast( round( diff [ 1 ]/ 1000 / 1000000.0, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' ) END AS "VALUE" , CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio FROM ( SELECT compare ( DATA, 86400 ) AS diff FROM ( SELECT count( 1 ) AS DATA FROM log ) )
```

- UV图表展示所选时间内有多少不同的ip发起了调用以及与前一天的趋势对比，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 1000 THEN concat( cast( diff [ 1 ] AS VARCHAR ), ' 次' ) WHEN diff [ 1 ] < 1000 * 1000 THEN concat( cast( round( diff [ 1 ]/ 1000, 1 ) AS VARCHAR ), ' 千次' ) WHEN diff [ 1 ] < 1000000000 THEN concat( cast( round( diff [ 1 ]/ 1000000.0, 1 ) AS VARCHAR ), ' 百万次' ) WHEN diff [ 1 ]/ 1000.0 < 1000000000 THEN concat( cast( round( diff [ 1 ]/ 1000 / 1000000.0, 1 ) AS VARCHAR ), ' 十亿次' ) ELSE concat( cast( round( diff [ 1 ]/ 1000.0 / 1000 / 1000 / 1000, 1 ) AS VARCHAR ), ' 万亿次' ) END AS "VALUE" , CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio FROM ( SELECT compare ( DATA, 86400 ) AS diff FROM ( SELECT count( DISTINCT CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) AS "DATA" FROM log ) )
```

- 流入流量图表展示请求流量大小以及与前一天的趋势对比，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( diff [ 1 ] AS VARCHAR ), ' B' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024, 1 ) AS VARCHAR ), ' KB' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024.0 / 1024, 1 ) AS VARCHAR ), ' MB' ) WHEN diff [ 1 ]/ 1024.0 < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024, 1 ) AS VARCHAR ), ' GB' ) ELSE concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024 / 1024, 1 ) AS VARCHAR ), ' TB' ) END AS "VALUE" , CASE WHEN diff [ 2 ]= 0 THEN 0 ELSE round( diff [ 3 ]- 1, 2 ) END AS ratio FROM ( SELECT compare ( "DATA", 86400 ) AS diff FROM ( SELECT COALESCE ( sum( request_length ), 0 ) AS "DATA" FROM log ) )
```

- 网络in带宽峰值展示所选时间段内请求流量带宽峰值以及与前一天同时段的趋势对比，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( round( diff [ 1 ], 2 ) AS VARCHAR ), ' B/s' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024, 1 ) AS VARCHAR ), ' KB/s' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN
```



```
concat( cast( round( diff [ 1 ]/ 1024.0 / 1024, 1 ) AS VARCHAR ), ' MB/s' )  WHEN diff [ 1 ]/ 1024.0 < 1024 * 1024 * 1024  THEN  concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024, 1 ) AS VARCHAR ), ' GB/s' )  ELSE concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024 / 1024, 1 ) AS VARCHAR ), ' TB/s' )  END AS  "VALUE"  ,  CASE  WHEN diff [ 2 ]= 0 THEN  0 ELSE round( diff [ 3 ]- 1, 2 )  END AS ratio  FROM  (  SELECT  compare ( "DATA", 86400 ) AS diff  FROM  (  SELECT COALESCE  ( max( "DATA" ), 0 ) AS "DATA"  FROM  ( SELECT TIME_FLOOR(__time,'PT1M') AS dt, sum( request_length )/ 60.0 AS "DATA" FROM log GROUP BY dt  LIMIT 10000 ))  )
```

- 网络out带宽峰值展示所选时间段内响应流量带宽峰值以及与前一天同时段的趋势对比，所关联的查询分析语句如下所示：

```
SELECT CASE WHEN diff [ 1 ] < 102 THEN concat( cast( round( diff [ 1 ], 2 ) AS VARCHAR ), ' B/s' ) WHEN diff [ 1 ] < 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024, 1 ) AS VARCHAR ), ' KB/s' ) WHEN diff [ 1 ] < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024.0 / 1024, 1 ) AS VARCHAR ), ' MB/s' ) WHEN diff [ 1 ]/ 1024.0 < 1024 * 1024 * 1024 THEN concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024, 1 ) AS VARCHAR ), ' GB/s' ) ELSE concat( cast( round( diff [ 1 ]/ 1024.0 / 1024 / 1024 / 1024, 1 ) AS VARCHAR ), ' TB/s' ) END AS "value", case when diff [ 2 ]= 0 then 0 else round( diff [ 3 ]- 1, 2 ) END AS "ratio" FROM ( SELECT compare ( "DATA", 86400 ) AS diff FROM ( SELECT COALESCE ( max( bytes_out ), 0 ) AS "DATA" FROM ( SELECT time_ceil( __time,'PT1M') AS dt, sum( body_bytes_sent )/ 60.0 AS bytes_out FROM log GROUP BY dt LIMIT 10000 )))
```

- **流量带宽趋势**图展示流入流量及流出流量随时间的变化情况，所关联的查询分析语句如下所示：

```
SELECT  TIME_FORMAT( MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 600000)), 'HH:mm' ) AS dt,
round( sum( request_length )/ 1024.0 / 600, 2 ) AS "流入流量(KB/s)",
round( sum( body_bytes_sent )/ 1024.0 / 600, 2 ) AS "流出流量(KB/s)"  where  request_length is not null  GROUP BY  TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 600000)  ORDER BY  dt  LIMIT 1000
```

- **PV/UV趋势**图展示访问次数变化情况，所关联的查询分析语句如下所示：

```
SELECT  TIME_FORMAT(MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 3600000)) , 'HH:mm' ) AS dt,  count( 1 ) AS PV,
APPROX_COUNT_DISTINCT (CASE WHEN sip = '-' THEN remote_ip ELSE sip  END) AS UV
FROM log  GROUP BY  TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 3600000)  ORDER BY  dt  LIMIT 1000
```

- **访问状态分布**图展示不同请求响应的错误码随时间变化情况，所关联的查询分析语句如下所示：

```
SELECT  TIME_FORMAT(MILLIS_TO_TIMESTAMP( TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 3600000)) , 'HH:mm' ) AS dt,  count( 1 ) AS cnt,
concat( cast( "response_code" / 100 AS VARCHAR ), 'XX' ) AS "status"  GROUP BY TIMESTAMP_TO_MILLIS(__time) - MOD(TIMESTAMP_TO_MILLIS(__time) , 3600000) , "response_code" / 100  ORDER BY  dt DESC  LIMIT 10000
```

- **访问来源**图展示访问IP来源再国内的省份分布，所关联的查询分析语句如下所示：

```
SELECT  ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS country,
count( 1 ) AS "访问次数"  where ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END) = '中国'  GROUP BY  country
```

- **流入流量来源（中国）**图展示流入流量地区在国内的分布情况，所关联的查询分析语句如下所示：

```
SELECT  ip_to_province (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,
round( sum( request_length )/ 1024.0 / 1024, 4 ) AS "流入流量(MB)"  where ip_to_country(CASE WHEN sip = '-' THEN remote_ip ELSE sip END) = '中国'  GROUP BY  region
```

- 流入流量来源（世界）图展示流入流量地区在世界的分布情况，所关联的查询分析语句如下所示：

```
SELECT  ip_to_country (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS region,
round( sum( request_length )/ 1024.0 / 1024, 4 ) AS "流入流量(MB)"  where  request_length is not null  GROUP BY  region
```

- **来源网络提供商**图展示各个来源网络提供商的占比，所关联的查询分析语句如下所示：

```
SELECT  ip_to_provider (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) AS provider,
round( sum( request_length )/ 1024.0 / 1024.0, 3 ) AS mb_in  GROUP BY  provider
HAVING  ip_to_provider (CASE WHEN sip = '-' THEN remote_ip ELSE sip END) != '*'  ORDER BY  mb_in DESC  LIMIT 10
```


- **访问域名**图展示被访问域名的访问次数，所关联的查询分析语句如下所示：
```
SELECT http_host, count( 1 ) AS "被访问次数" GROUP BY http_host ORDER BY "被访问次数" DESC LIMIT 30
```
- **响应最慢的URL**图展示响应时间最慢的URL，所关联的查询分析语句如下所示：
```
SELECT http_host AS "网站",url_extract_path (COALESCE ( url, '/' )) AS URL,sum( request_time )/ count( 1 ) AS "响应时间(毫秒)",count( 1 ) AS "访问次数" GROUP BY http_host, url ORDER BY "响应时间(毫秒)" DESC LIMIT 100
```
- 访问最多的客户端图展示所选时间段内访问次数最多ip所属地区，所关联的查询分析语句如下所示：
```
SELECT ip AS "客户端", client AS "地理网络", concat( cast( (CASE WHEN pv IS NULL THEN 0 ELSE pv END) AS VARCHAR ), ' (', cast( case when head_pv = 'null' then 0 else (case when head_pv > 0 then head_pv else 0 end) end AS VARCHAR ), '/', cast( case when get_pv = 'null' then 0 else (case when get_pv > 0 then get_pv else 0 end) end AS VARCHAR ), '/', cast( case when put_pv = 'null' then 0 else (case when put_pv > 0 then put_pv else 0 end) end AS VARCHAR ), '/', cast( case when post_pv = 'null' then 0 else (case when post_pv > 0 then post_pv else 0 end) end AS VARCHAR ), '/', cast( case when delete_pv = 'null' then 0 else (case when delete_pv > 0 then delete_pv else 0 end) end AS VARCHAR ), '/', ')' ) AS "PV (Head, Get, Put, Post, Delete方法)", error_count AS "错误访问次数" FROM ( SELECT ip, client, sum( CASE WHEN "method" = 'PUT' AND "status" < 400 THEN pv ELSE 0 END ) AS put_pv, sum( CASE WHEN "method" = 'GET' AND "status" < 400 THEN pv ELSE 0 END ) AS get_pv, sum( CASE WHEN "method" = 'POST' AND "status" < 400 THEN pv ELSE 0 END ) AS post_pv, sum( CASE WHEN "method" = 'DELETE' AND "status" < 400 THEN pv ELSE 0 END ) AS delete_pv, sum( CASE WHEN "method" = 'HEAD' AND "status" < 400 THEN pv ELSE 0 END ) AS head_pv, sum( throughput ) AS throughput, sum( pv ) AS pv, sum( CASE WHEN "status" < 400 THEN 1 ELSE 0 END ) AS error_count FROM ( SELECT CASE WHEN sip = '-' THEN remote_ip ELSE sip END AS ip, "method", CASE WHEN ip_to_country ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )= '上海' THEN '中国上海' WHEN ip_to_province ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )= '*' THEN '未知IP' WHEN ip_to_provider ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )= '内网IP' THEN '内网IP' ELSE concat( ip_to_country ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ), '/', ip_to_province ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ), '/', CASE WHEN ip_to_city ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )= '*' THEN ' ' ELSE ip_to_city ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END ) END, ' ', ip_to_provider ( CASE WHEN sip = '-' THEN remote_ip ELSE sip END )) END AS client, sum( CASE WHEN "response_code" < 400 THEN 1 ELSE 0 END ) AS pv, round( sum( request_length )/ 1024.0 / 1024, 1 ) AS throughput, "response_code" AS "status" FROM log GROUP BY ip, client, "method", "response_code" ORDER BY pv DESC, client, "method" LIMIT 1000 ) GROUP BY ip, client ORDER BY pv DESC ) LIMIT 100
```

7.3.3.18 采集诊断仪表盘模板

7.3.3.18.1 ICAgent 采集监控

云日志服务支持ICAgent采集监控仪表盘模板，用于展示采集文件数、采集机器数/同比昨天、采集文件分布、采集机器数等图表。

前提条件

在LTS控制台配置中心页面的“ICAgent采集开关”页签，开启ICAgent诊断开关，请参考**设置ICAgent日志采集开关**。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“采集诊断仪表盘模板>ICAgent采集监控”仪表盘，查看图表详情。

----结束

ICAgent采集监控仪表盘中的过滤器说明如下所示：

- 日志组ID，所关联的查询分析语句如下所示：
```
select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by loggroup
```
- 日志流ID，所关联的查询分析语句如下所示：
```
select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by logstream
```

ICAgent采集监控仪表盘中的重要图表说明如下所示：

- 原始数据流量图展示原始数据流量的变化情况，所关联的查询分析语句如下所示：
```
SELECT
    case
        when diff [ 1 ] is null then '0'
        when diff [ 1 ] > 1024 and diff [ 1 ] <= 1024*1024 then concat(round(diff [ 1 ]*1.0/1024,4),' KB')
        when diff [ 1 ] > 1024*1024 and diff [ 1 ] < 1024*1024*1024 then  concat(round(diff [ 1 ]*1.0/1024/1024,4),' MB')
        when diff [ 1 ] > 1024*1024*1024 and diff [ 1 ] < 1024*1024*1024*1024 then  concat(round(diff [ 1 ]*1.0/1024/1024/1024,4),' GB')
        when diff [ 1 ] > 1024*1024*1024*1024 then  concat(round(diff [ 1 ]*1.0/1024,4),' TB')
        else concat(round(diff [ 1 ]*1.0,2),' B')
    END AS "原始流量",
    case
        when diff [ 3 ] is null then '昨日无数据'
        else round(diff [ 3 ] - 1,2)
    END AS "同比昨日"
FROM
    (
    SELECT
        report_topic,
        compare ( traffic, 86400 ) AS diff
    FROM
        ( SELECT report_topic, sum( read_bytes ) AS traffic FROM log WHERE report_topic = 'icagent_profile' GROUP BY report_topic )
GROUP BY
    report_topic)
```
- **采集文件数**图展示采集文件的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "采集文件数" , case when diff[3] is not null then round(diff[3] -1 ,2) else '昨日无数据' end as "同比昨天" from  (select compare(uv,86400) as diff from (select report_topic,count(distinct concat(file_name,host_ip)) as uv from log where report_topic = 'icagent_profile' group by report_topic) group by report_topic)
```
- **采集机器数/同比昨天**图展示采集机器数/同比昨天的变化情况，所关联的查询分析语句如下所示
```
select diff[1] as "采集机器数" ,  case when diff[3] is not null then round(diff[3] -1 ,2) else '昨日无数据' end as "同比昨天" from (select compare(uv,86400) as diff from (select report_topic,count(distinct host_ip) as uv from log where report_topic = 'icagent_profile' group by report_topic) group by report_topic)
```
- **数据发送流量**图展示发送流量情况，所关联的查询分析语句如下所示：
```
SELECT
    "time",
        case
        when traffic is null then 0
        else round(traffic*1.0/1024/1024,2)
    END AS "发送流量 MB"
    from
    (SELECT
        time_format( time_floor ( __time, 'PT5M' ), 'yyyy-MM-dd HH:mm' ) AS "time",
        sum( read_bytes ) AS "traffic"
    FROM
        log
    WHERE
        report_topic = 'icagent_profile'
    GROUP BY
        "time")
```


- **ICAgent写入次数**图展示ICAgent写入次数变化情况，所关联的查询分析语句如下所示：
```
select time_floor(__time,'PT5M') as \"time\",sum(read_count) as \"写入次数\" where report_topic = 'icagent_profile'  group by \"time\"
```
- **采集机器数**图展示采集机器的变化情况，所关联的查询分析语句如下所示：
```
select time_floor(__time,'PT5M') as "time" , count(distinct host_ip) as "采集机器数" where report_topic = 'icagent_profile' group by "time"
```
- **采集文件分布**图展示采集文件的情况，所关联的查询分析语句如下所示：
```
SELECT
      file_name AS "采集路径",
      host_ip AS "IP",
      case
         when traffic is null then '0'
         when traffic > 1024 and traffic <= 1024*1024 then concat(round(traffic*1.0/1024,2),' KB')
         when traffic > 1024*1024 and traffic < 1024*1024*1024 then
concat(round(traffic*1.0/1024,2),' MB')
         when traffic > 1024*1024*1024 and traffic < 1024*1024*1024*1024 then
concat(round(traffic*1.0/1024,2),' GB')
         when traffic > 1024*1024*1024*1024 then  concat(round(traffic*1.0/1024,2),' TB')
      else concat(round(traffic*1.0,2),' B')
   END AS "采集流量"
   FROM
   (SELECT
      file_name,
      host_ip,
      sum( read_bytes ) AS "traffic"
   WHERE
      "report_topic" = 'icagent_profile'
   GROUP BY
      file_name,
      host_ip)
```

7.3.3.18.2 ICAgent 整体状态

云日志服务支持展示ICAgent整体状态仪表盘模板，用于展示活跃ICAgent数、CPU趋势、ICAgent整体状态等图表。

前提条件

在LTS控制台配置中心页面的“ICAgent采集开关”页签，开启ICAgent诊断开关，请参考**设置ICAgent日志采集开关**。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“采集诊断仪表盘模板>ICAgent整体状态”仪表盘，查看图表详情。

----结束

ICAgent整体状态仪表盘中的重要图表说明如下所示：

- **活跃ICAgent数**图展示常用日志流的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "活跃ICAgent数",case when diff[2] is not null then diff[2] else '昨日无数据' end as "昨日活跃ICAgent数" from  (select report_topic,compare(uv,86400) as diff from (SELECT report_topic,COUNT(DISTINCT ip) as uv FROM log where report_topic = 'icagent_status' group by report_topic) group by report_topic)
```


- **发送延迟/次数趋势**图展示延迟/次数变化情况，所关联的查询分析语句如下所示：
```
select time_floor(__time,'PT5M') as \"time\" , sum(\"metric.lts_cost.below_100_ms\") as \"below_100_ms\" , sum(\"metric.lts_cost.100to500ms\") as \"100to500ms\" , sum(\"metric.lts_cost.500msto1s\") as \"500msto1s\" , sum(\"metric.lts_cost.1sto10s\") as \"1sto10s\", sum(\"metric.lts_cost.10ston\") as \"10ston\" from log where \"report_topic\" = 'icagent_status' group by \"time\"
```
- **运行状态分布**图展示运行状态的变化情况，所关联的查询分析语句如下所示：
```
select status,count(DISTINCT ip) as pv from log where report_topic = 'icagent_status' group by status
```
- **CPU趋势**图展示CPU占用率的变化情况，所关联的查询分析语句如下所示：
```
select ip,time_floor(__time,'PT5M') as \"time\",avg(\"metric.cpu_usage\") as \"CPU占用率\" from log where report_topic = 'icagent_status' and \"metric.cpu_usage\" is not null group by \"time\",ip order by \"time\"
```
- ICAgent整体状态图展示主机的整体情况，所关联的查询分析语句如下所示：
```
select host_name as "主机名",ip as "IP" , version as "版本号" , os as "操作系统" , time_format(MILLIS_TO_TIMESTAMP(ANY_VALUE("metric.start_time")),'yyyy/MM/dd HH:mm:ss ZZZ') as "启动时间",avg("metric.cpu_usage") as "CPU",avg("metric.mem_used")*1.0 as "内存(MB)",status as "运行状态" where report_topic = 'icagent_status' group by host_name,ip,version,os,status
```

7.3.3.18.3 ICAgent 异常监控

云日志服务支持ICAgent异常监控仪表盘模板，用于展示关键错误数、丢弃超大行、请求LTS失败、文件超过上限问题数等图表。

前提条件

在LTS控制台配置中心页面的“ICAgent采集开关”页签，开启ICAgent诊断开关，请参考**设置ICAgent日志采集开关**。

分析网站访问情况

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“仪表盘 ”。

步骤3 在仪表盘模板下方，选择“采集诊断仪表盘模板>ICAgent异常监控”仪表盘，查看图表详情。

----结束

ICAgent异常监控仪表盘中的过滤器说明如下所示：

- 日志组ID，所关联的查询分析语句如下所示：
```
select loggroup from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by loggroup limit 10000
```
- 日志流ID，所关联的查询分析语句如下所示：
```
select logstream from log where report_topic = 'icagent_profile' or report_topic = 'icagent_alarm' group by logstream limit 10000
```

ICAgent异常监控仪表盘中的重要图表说明如下所示：

- **关键错误数**图展示关键错误的变化情况，所关联的查询分析语句如下所示：
```
select diff[1] as "错误数", case when diff[3] is not null then round(diff[3] - 1 , 2) else '昨日无数据' end as "错误数对比昨天" from (select report_topic,compare(pv , 86400) as diff from (select report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' group by report_topic) group by report_topic)
```
- **丢弃超大行**图展示丢弃超大行的变化情况，所关联的查询分析语句如下所示
```
select diff[1] as "丢弃行数" , case when diff[3] is not null then round(diff[3] - 1 , 2) else '昨日无数据' end as "丢弃行数对比昨天" from (select report_topic,compare(pv , 400) as diff from (select
```



```
report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type =
'DISCARD_BIG_LINE' group by report_topic) group by report_topic)
```

- **请求LTS失败**图展示请求LTS失败的变化情况，所关联的查询分析语句如下所示：

```
select diff[1] as "请求失败数" , case when diff[3] is not null then round(diff[3] - 1 , 2) else '昨日无数
据' end as "请求失败数对比昨天" from (select report_topic,compare(pv , 86400) as diff from (select
report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and alarm_type =
'HTTP_REQUEST_ALARM' group by report_topic) group by report_topic)
```

- **文件超过上限**问题数图展示文件超过上限的变化情况，所关联的查询分析语句如下所示：

```
select diff[1] as "文件超过上限问题数" ,case when diff[3] is not null then round(diff[3] - 1 , 2) else '昨
日无数据' end as "文件超过上限问题数对比昨天" from (select  report_topic,compare(pv , 86400) as diff
from (select  report_topic,count(1) as pv from log where report_topic = 'icagent_alarm' and
alarm_type = 'COLLECT_FILE_EXCEED' group by report_topic) group by report_topic)
```

- **关键错误数(必需处理)**图展示关键错误的整体情况，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)),'yyyy/MM/dd
HH:mm:ss ZZ') as "最近发生时间" , loggroup as "日志组ID" ,logstream as "日志流ID" ,  host_ip as "IP" ,
alarm_type as "告警类型", os as "系统类型" , alarm_message as "告警详情" where report_topic =
'icagent_alarm' group by loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **丢弃超大行详情**图展示丢弃超大行详情，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)),'yyyy/MM/dd
HH:mm:ss ZZ') as "最近发生时间" , loggroup as "日志组ID" ,logstream as "日志流ID" ,  host_ip as "IP" ,
alarm_type as "告警类型", os as "系统类型" , alarm_message as "告警详情" where report_topic =
'icagent_alarm' and alarm_type = 'DISCARD_BIG_LINE' group by
loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **请求LTS失败详情**图展示发生时间、告警类型、告警详情等信息，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)),'yyyy/MM/dd
HH:mm:ss ZZ') as "最近发生时间" , loggroup as "日志组ID" ,logstream as "日志流ID" ,  host_ip as "IP" ,
alarm_type as "告警类型", os as "系统类型" , alarm_message as "告警详情" where report_topic =
'icagent_alarm'  and alarm_type = 'HTTP_REQUEST_ALARM' group by
loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```

- **文件超过上限问题数详情**图展示发生时间、告警类型、告警详情等信息，所关联的查询分析语句如下所示：

```
select TIME_FORMAT(MILLIS_TO_TIMESTAMP(ANY_VALUE(report_time/1000000)),'yyyy/MM/dd
HH:mm:ss ZZ') as "最近发生时间" , loggroup as "日志组ID" ,logstream as "日志流ID" ,  host_ip as "IP" ,
alarm_type as "告警类型", os as "系统类型" , alarm_message as "告警详情" where report_topic =
'icagent_alarm'  and alarm_type = 'COLLECT_FILE_EXCEED' group by
loggroup,logstream,host_ip,alarm_type,os,alarm_message limit 10000
```


8 日志告警

8.1 日志告警概述

云日志服务支持创建搜索分析类型、关键词统计类型、SQL统计类型的日志告警规则，根据设置的告警规则触发告警，可以在告警列表查看上报的告警详情，实时监控服务运行状态。或者通过告警行动规则将上报告警以短信，手机，邮件等多种形式发送告警通知，方便用户及时处理告警问题。

8.2 配置日志告警规则

LTS支持对日志流中的日志数据进行关键词统计，通过设置告警规则，监控日志中的关键词，通过在一定时间段内，统计日志中关键字出现的次数，实时监控服务运行状态。目前每个账号最多可以创建关键词告警与SQL告警共200个。

云日志服务支持将日志数据进行结构化，通过配置SQL告警规则，定时查询结构化数据，当且仅当条件表达式返回为true的时候，将告警进行上报，用户可以在LTS控制台查看SQL告警。每条SQL告警规则可以关联1到3个图表，每个图表包含一条查询某个日志流的SQL查询语句。

前提条件

已创建日志组、日志流。

创建单个搜索分析告警（管道符方式-邀测）

说明

目前此功能在邀测中，暂不支持申请开通。

云日志服务支持将日志数据进行结构化，通过新SQL引擎配置告警规则，支持使用管道符（搜索|分析），定时查询结构化数据，当且仅当条件表达式返回为true的时候，将告警进行上报，用户可以在LTS控制台查看SQL告警。

步骤1 在云日志服务管理控制台，单击“日志告警”。

步骤2 在告警页面默认显示“告警列表”，单击“告警规则”切换至告警规则页面。

步骤3 单击“创建”，在界面右侧弹出“新建告警规则”页面。

步骤4 在“新建告警规则”页面，配置告警规则相关参数。

表 8-1 搜索分析告警条件填写说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称。名称只支持输入英文、数字、中文、中划线、下划线，且不能中划线、下划线开头或结尾。长度为 1-64个字符。 **说明** 告警创建完成后，支持修改规则名称，修改完成后，鼠标悬浮在规则名称上，显示修改后的规则名称和原始名称，不支持修改首次创建的原始名称。
	描述	对该规则进行简要描述。长度不能超过64个字符。
统计分析	统计类型	勾选搜索分析：使用新SQL引擎配置告警，支持使用管道符（搜索|分析）。
	查询条件（支持添加3条查询语句。）	日志组名称：选择已创建的日志组。
		日志流名称：选择已创建的日志流。 **说明** 当日志组下有多个日志流时，支持选择多个日志流，即可批量创建关键词告警。
		查询时间：指定语句的查询周期。查询语句的时间范围：从当前时间往前推一个周期。例如：查询时间设置为1小时，当前时间为9:00，则查询语句的时间范围为8:00-9:00。 ● 如果查询时间单位为分钟，则取值范围是1-60； ● 如果查询时间单位为小时，则取值范围是1-24。
		查询语句：格式为搜索语句 | SQL分析语句，LTS会根据设置的语句对日志流中的日志进行监控。

参数类别	参数名称	参数说明
	校验规则	当满足条件表达式（ ）时，告警级别为（紧急、重要、次要、提示）And（ ）次统计周期内，至少满足（ ）次以上条件触发告警。 输入具体的条件表达式，当条件表达式返回为true的时候，产生告警，否则不产生告警。 单击+增加条件表达式（or），最多支持增加20条。 单击 删除条件表达式。 **说明** ● 条件表达式支持中文。 ● 条件表达式不支持纯数字，不支持以数字开头的。 ● 统计周期次数指上面设置的统计周期；满足条件次数指设置的条件表达式。配置的统计周期次数须大于等于满足触发条件次数。 ● 统计周期次数最小值为1，最大值为10。 条件表达式支持的基础语法和多表组合语法。 ● 基础语法： – 基础运算符：支持加（+）、减（-）、乘（*）、除（/）、取模运算（%）。 示例：x * 10 + y > 100。 – 比较运算符：支持大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。示例：x >= 100。 – 逻辑运算符：支持与（&&）、或（||）。示例：x > 0 && y < 200。 – 取反前缀：支持取反前缀（!）。 示例：!(x < 1 && x > 100)。 – 数值常量：支持数值常量，并作为64位浮点数处理。示例：x > 10。 – 字符串常量：支持字符串常量（"字符串"），例如"string"。示例：str =="string"。 – 布尔常量：支持布尔常量(true、false)。示例：(x < 100)!=true。 – 括号：支持使用括号改变计算的优先级。示例：x *(y + 10)<200。 – contains函数：支持使用contains函数判断是否包含子串，例如contains(str, "hello")返回true则表示str中包含hello子串。 ● 多表组合语法： – 基础运算符：（+-*/%）。 – 比较运算符：大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。 – 逻辑运算符：与（&&）、或（||）。

参数类别	参数名称	参数说明
		– 取反前缀（!）。 – contains函数。 – 括号（ ）。
高级设置	统计周期	条件表达式查询的频率可以设置为： ● 每小时：表示整点小时查询。 ● 每天：需要指定几点整查询。 ● 每周：需要指定周几的几点整查询。 ● 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10..... **说明** 当查询时间大于1小时，固定间隔时间最小取值为5分钟。 ● CRON表达式：CRON表达式的最小精度为分钟，格式为24小时制，示例如下： – 0/10 * * * *从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。例如：当前时间为16:37，下一次查询时间为16:50。 – 0 0/5 * * *从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。例如：当前时间为16:37，下一次查询时间为20:00。 – 0 14 * * *每天14:00查询一次。 – 0 0 10 * *每月10日00:00查询一次。
	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。
	通知场景	● 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 ● 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。

参数类别	参数名称	参数说明
	告警行动规则	请从下拉列表中选择已创建的告警行动规则。 若没有，请单击右侧“创建告警行动规则”，详细操作请见**创建告警行动规则**。
	语言	发送告警的语言，支持中文（简体）和英文。

步骤5 单击“确定”，完成对搜索分析告警规则的创建。

说明

告警规则创建完成后，告警状态默认显示“已开启”。关闭告警规则后，告警状态显示“已关闭”，临时关闭告警后，告警状态显示“临时关闭到2023/05/30 16:21:24.000 GMT+08:00”。（临时关闭的时间仅供参考，请以设置临时关闭告警的时间为准）

当开启告警规则且关联日志流满足告警规则时，会触发告警；当关闭告警规则时，即使有满足该告警规则的情况，也不会触发告警。

----结束

创建关键词告警规则

步骤1 在云日志服务管理控制台，单击“日志告警”。

步骤2 在告警页面默认显示“告警列表”，单击“告警规则”切换至告警规则页面。

步骤3 单击“创建”，在界面右侧弹出“新建告警规则”页面。

步骤4 在“新建告警规则”页面，配置告警规则相关参数。

表 8-2 关键词告警条件填写说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称。名称只支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾。长度为 1-64个字符。 **说明** 告警创建完成后，支持修改规则名称，修改完成后，鼠标悬浮在规则名称上，显示修改后的规则名称和原始名称，不支持修改首次创建的原始名称。
	描述	对该规则进行简要描述。长度不能超过64个字符。
统计分析	统计类型	勾选关键词统计：适用于使用关键词搜索配置日志告警的场景。
	查询条件	日志组名称：选择已创建的日志组。
		日志流名称：选择已创建的日志流。 **说明** 当日志组下有多个日志流时，支持选择多个日志流，即可批量创建关键词告警。

参数类别	参数名称	参数说明
		查询时间：指定语句的查询周期。查询语句的时间范围：从当前时间往前推一个周期。例如：查询时间设置为1小时，当前时间为9:00，则查询语句的时间范围为8:00-9:00。 ● 如果查询时间单位为分钟，则取值范围是1-60； ● 如果查询时间单位为小时，则取值范围是1-24。
		关键词：LTS会根据设置的关键词对日志流中的日志进行监控。关键词支持精确匹配和模糊匹配，区分大小写，输入长度不超过1024个字符。
	检测规则	配置触发条件，即满足该条件时，会触发告警。 匹配条数：当关键词搜索结果的日志条数达到设定的条数时，会触发告警。支持大于（>）、大于等于（>=）、小于（<）、小于等于（<=）4种比较运算符。 ● 单击+增加条件表达式（or），最多支持增加20条。 ● 单击 删除条件表达式。 统计周期次数指高级设置的统计周期；满足条件次数指设置的关键词。配置的统计周期次数须大于等于满足触发条件次数。 **说明** ● 触发告警级别包括“紧急”、“重要”、“次要”、“提示”，默认“紧急”。 ● 统计周期次数最小值为1，最大值为10。

参数类别	参数名称	参数说明
高级设置	统计周期	条件表达式查询的频率可以设置为： ● 每小时：表示整点小时查询。 ● 每天：需要指定几点整查询。 ● 每周：需要指定周几的几点整查询。 ● 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10..... **说明** 当查询时间大于1小时，固定间隔时间最小取值为5分钟。 ● CRON表达式：CRON表达式的最小精度为分钟，格式为24小时制，示例如下： – 0/10 * * * *从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。例如：当前时间为16:37，下一次查询时间为16:50。 – 0 0/5 * * *从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。例如：当前时间为16:37，下一次查询时间为20:00。 – 0 14 * * *每天14:00查询一次。 – 0 0 10 * *每月10日00:00查询一次。
高级设置	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。
高级设置	通知场景	● 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 ● 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
高级设置	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。
高级设置	告警行动规则	请从下拉列表中选择已创建的告警行动规则。 若没有，请单击右侧“创建告警行动规则”。
高级设置	语言	发送告警的语言，支持中文（简体）和英文。

步骤5 单击“确定”，完成对关键词告警规则的创建。

说明

告警规则创建完成后，告警状态默认显示“已开启”。关闭告警规则后，告警状态显示“已关闭”，临时关闭告警后，告警状态显示“临时关闭到2023/05/30 16:21:24.000 GMT+08:00”。（临时关闭的时间仅供参考，请以设置临时关闭告警的时间为准）

当开启告警规则且关联日志流满足告警规则时，会触发告警；当关闭告警规则时，即使有满足该告警规则的情况，也不会触发告警。

----结束

创建 SQL 告警规则

步骤1 在云日志服务管理控制台，单击“日志告警”。

步骤2 在告警页面默认显示“告警列表”，单击“告警规则”切换至告警规则页面。

步骤3 单击“创建”，在界面右侧弹出“新建告警规则”页面。

步骤4 在“新建告警规则”页面，配置告警规则相关参数。

表 8-3 SQL 告警条件填写说明

参数类别	参数名称	参数说明
基本信息	规则名称	告警规则的名称。名称只支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾。长度为 1-64个字符。 **说明** 告警创建完成后，支持修改规则名称，修改完成后，鼠标悬浮在规则名称上，显示修改后的规则名称和原始名称。不支持修改首次创建的原始名称。
	描述	对该规则进行简要描述。长度不能超过64个字符。
统计分析	统计类型	勾选SQL统计：使用SQL分析配置告警。

参数类别	参数名称	参数说明
	相关图表	有两种添加方式：直接添加和从图表导入 • **直接添加**：单击“直接添加”，可选择日志组、日志流。具体的参数配置信息如下： 日志组名称：日志组的名称，必选项。 日志流名称：日志组下的日志流名称，必选项。 **说明** 若所选日志流未配置结构化规则，请先**配置结构化**。 查询时间：当前所选日志的查询时间，可选项。查询时间（1 ~ 60分钟/1 ~ 24小时），单位为分钟或小时。 查询语句：可视化查询语句，必填项。具体请参考**SQL语法查询**。 • **从图表导入**：单击 ＋从图表导入 ，进入“添加可视化图表”页面，选择对应日志组、日志流下的可视化图表，单击“确定”。若该日志流下没有图表或没有所需的图表，单击界面上的“前往添加图表”，进入可视化界面，设置完成后单击“保存并返回”返回到告警规则界面，自动打开创建规则弹框，填充新创建的图表及图表的查询语句。 可以指定图表的查询时间（1 ~ 60分钟/1 ~ 24小时），单位为分钟或小时，每个图表最多可以查询最近一天的数据，当统计周期选择1~4分钟时，图表查询时间不能超过1小时。 若想添加多个图表，可单击 ＋从图表导入 继续添加。 **说明** – 单击 跳转到日志流的可视化查看详情界面。 – 单击 删除该直接添加的图表。 – 单击“预览”可查看可视化分析后的数据。必须要执行“预览”，否则将无法保存该告警规则。 – 最多支持添加3个图表。 – 图表不能为空，且图表中的sql查询语句不能为空。

参数类别	参数名称	参数说明
	检测规则	输入具体的条件表达式，当条件表达式返回为true的时候，产生告警，否则不产生告警。 **说明** • 条件表达式支持中文。 • 条件表达式不支持纯数字，不支持以数字开头的。 • 统计周期次数指上面设置的统计周期；满足条件次数指设置的条件表达式。配置的统计周期次数须大于等于满足触发条件次数。 • 触发告警级别包括“紧急”、“重要”、“次要”、“提示”，默认“紧急”。 • 统计周期次数最小值为1，最大值为10。 ● 单击+增加条件表达式（or），最多支持增加20条。 ● 单击 删除条件表达式。 条件表达式支持的基础语法和多表组合语法。 ● 基础语法： – 基础运算符：支持加（+）、减（-）、乘（*）、除（/）、取模运算（%）。 示例：x * 10 + y > 100。 – 比较运算符：支持大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。示例：x >= 100。 – 逻辑运算符：支持与（&&）、或（||）。示例：x > 0 && y < 200。 – 取反前缀：支持取反前缀（!）。 示例：!(x < 1 && x > 100)。 – 数值常量：支持数值常量，并作为64位浮点数处理。示例：x > 10。 – 字符串常量：支持字符串常量（"字符串"），例如"string"。示例：str =="string"。 – 布尔常量：支持布尔常量(true、false)。示例：(x < 100)!=true。 – 括号：支持使用括号改变计算的优先级。示例：x *(y + 10)<200。 – contains函数：支持使用contains函数判断是否包含子串，例如contains(str, "hello")返回true则表示str中包含hello子串。 ● 多表组合语法： – 基础运算符：（+-*/%）。 – 比较运算符：大于（>）、大于等于（>=）、小于（<）、小于等于（<=）、等于（==）、不等于（!=）。 – 逻辑运算符：与（&&）、或（||）。 – 取反前缀（!）。

参数类别	参数名称	参数说明
		– contains函数。 – 括号（ ）。
高级设置	统计周期	条件表达式查询的频率可以设置为： ● 每小时：表示整点小时查询。 ● 每天：需要指定几点整查询。 ● 每周：需要指定周几的几点整查询。 ● 固定间隔：自定义间隔周期，需要指定1-60分钟/1-24小时。例如：当前时间为9:00，固定间隔设置为5分钟，则第一次查询时间为9:00，第二次查询时间为9:05，第三次查询时间为9:10..... **说明** 当查询时间大于1小时，固定间隔时间最小取值为5分钟。 ● CRON表达式：CRON表达式的最小精度为分钟，格式为24小时制，示例如下： – 0/10 * * * *从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。例如：当前时间为16:37，下一次查询时间为16:50。 – 0 0/5 * * *从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。例如：当前时间为16:37，下一次查询时间为20:00。 – 0 14 * * *每天14:00查询一次。 – 0 0 10 * *每月10日00:00查询一次。
高级设置	恢复策略	配置恢复策略，即满足该策略时，会发送告警恢复通知。 配置的最近统计周期次数内，如果不满足触发条件且开启恢复时通知开关，则会发送恢复告警通知。 最近统计周期次数最小值为1，最大值为10。
高级设置	通知场景	● 告警触发时：用于发送触发告警通知。开启该按钮，当满足触发条件时，会发送告警通知；未开启该按钮，当满足触发条件时，不会发送告警通知。 ● 告警恢复时：用于发送恢复告警通知。开启该按钮，当满足恢复策略时，会发送恢复告警通知；未开启该按钮，当满足恢复策略时，不会发送恢复告警通知。
高级设置	通知频率	支持选择立即通知、每5分钟、每10分钟、每15分钟、每30分钟、每1小时、每3小时、每6小时发送告警。 立即通知指只要产生告警就发送通知，每10分钟指的是两次通知之间最小时间间隔为10分钟，可避免告警轰炸。
高级设置	告警行动规则	请从下拉列表中选择已创建的告警行动规则。 若没有，请单击右侧“创建告警行动规则”，详细操作请见**创建告警行动规则**。

参数类别	参数名称	参数说明
高级设置	语言	发送告警的语言，支持中文（简体）和英文。

步骤5 单击“确定”。

----结束

创建多个告警规则

支持批量创建告警规则。

步骤1 在“告警规则”页面，批量导入告警规则。

1. 单击“导入”，进入导入告警规则页面。
2. 下载告警模板到本地填写完成。
3. 单击“选择文件”，选择本地填写好的文件。
4. 确认导入的规则信息无误后，单击“导入”。
5. 导入成功后，在规则列表下方显示告警规则明细。

步骤2 单击“批量编辑”，进入批量编辑告警规则页面。

步骤3 在基本配置下方，输入告警规则数量，单击“添加告警规则”。

或者单击“导入”，批量导入告警规则。

说明

在规则列表下方默认已有1个告警规则，最多支持再添加199个数量，因此支持同时添加200个告警规则。

步骤4 在规则列表下方，请参考**创建单个搜索分析告警（管道符方式-邀测）**、**创建关键词告警规则**、**创建SQL告警规则**设置告警规则，设置完成后，单击“提交”。

- 一个告警规则设置完成后，单击“应用于其他告警规则”即可将该告警规则复制到其他告警规则。
- 例如添加了4个告警规则，批量创建成功后，在告警规则页签下方，就会显示4条告警规则。

----结束

告警规则后续操作

- 支持对单个告警规则进行如下操作：

  修改告警规则：单击告警规则所在行后的 按钮，根据**表8-2**修改具体参数，支持修改规则名称，修改完成后，鼠标悬浮在规则名称上，显示修改后的规则名称和原始名称。不支持修改首次创建的原始名称。

  开启告警规则：单击告警规则所在行后的 按钮（关闭告警规则后，才会显示开启按钮），开启告警规则。

  关闭告警规则：单击告警规则所在行后的 按钮（开启告警规则后，才会显示关闭按钮），关闭告警规则。

临时关闭告警规则：单击告警规则所在行后的 按钮，设置临时关闭的截止时间。

复制告警规则：单击告警规则所在行后的按钮，复制告警规则。

删除告警规则：单击告警规则所在行后的 按钮，单击“确定”删除该规则。

- 勾选多个告警规则后，支持对多个告警进行批量操作：开启、关闭、临时关闭、取消临时关闭、告警恢复开启、告警恢复关闭、删除、导出。

8.3 配置日志告警行动规则

8.3.1 在 LTS 页面创建消息模板

消息模板是告警通知消息的固定格式，系统发送告警通知消息必须使用消息模板向订阅者发送。默认内置消息模板分别为关键词模板、keywords_template、sql模板和sql_template。不同协议的订阅者优先选择模板名称对应的协议模板，如果对应的协议模板不存在，则采用内置的消息模板。使用消息模板发送告警通知消息时，系统会自动将模板变量替换为告警规则中的内容。

创建消息模板

步骤1 在云日志服务管理控制台，单击“日志告警”，进入告警页面，选择“告警行动规则”。

说明

消息模板默认有以下内置模板，当您所选择的消息模板中未配置消息内容时，云日志服务默认使用内置模板。

- 关键词模板：关键词告警模板
- keywords_template：关键词告警英文模板
- sql模板：sql告警模板
- sql_template：sql告警英文模板

步骤2 在消息模板页签，单击“创建”，在界面右侧弹出的“创建消息模板”页面中，配置消息模板的相关参数。

表 8-4 配置消息模板参数

参数名称	说明	校验规则	样例
模板名称	消息模板的名称	输入内容只能是数字、字母、下划线、汉字、中划线，且不能以下划线、中划线等特殊符号开头和结尾。长度不能超过100个字符。	LTS-test
模板描述	对消息模板的描述	输入内容只能是数字、字母、下划线、汉字，且不能以下划线等特殊符号开头和结尾。长度不能超过1024个字符。	-

参数名称	说明	校验规则	样例
消息头语言	系统在发送消息时会默认添加消息头	• 中文（简体） • 英文	• 中文：“尊敬的用户...” • 英文：“Dear User...”
通知方式	消息的通知方式类型	• 邮件 • 短信 • HTTP/HTTPS • 钉钉 • 飞书 • 企业微信 • 语音 **说明** 语音和飞书的功能仅针对白名单用户提交工单申请使用。详细操作请参考**提交工单**。	-
主题	消息的主题	支持自定义主题名称和使用变量命名主题两种方式。主题名称长度不能超过512个字符。 仅邮件类型支持配置消息主题。	test

参数名称	说明	校验规则	样例
正文	消息的内容	**添加变量：** ● 规则原始名称：${event_name} ● 告警级别：${event_severity} ● 发生时间：${starts_at} ● 发生区域：${region_name} ● 华为云账号：${domain_name} ● 告警源：$event.metadata.resource_provider ● 资源类型：$event.metadata.resource_type ● 资源标识：${resources} ● 告警状态：$event.annotations.alarm_status ● 表达式：$event.annotations.condition_expression ● 当前值：$event.annotations.current_value ● 统计周期：${frequency} ● 规则名称：${event_name} ● 通知频率：$event.annotations.notification_frequency ● 日志组原始名称：$event.annotations.results[0].log_group_name ● 日志流原始名称：$event.annotations.results[0].log_stream_name ● 关键词告警支持的变量 1. 查询时间：$event.annotations.results[0].time 2. 查询日志：（日志长度最多2KB，超过2KB被截断丢弃）	${event_name} ${event_severity} ${starts_at} ${region_name}

参数名称	说明	校验规则	样例
		$event.annotations.results[0].raw_results 3. 查询URL：$event.annotations.results[0].url 4. 日志组/日志流名称：$event.annotations.results[0].resource_id **说明** 只支持添加首次创建的日志组/日志流原始名称，不支持添加修改后的日志组/日志流名称。 5. 日志流的企业项目ID：$event.annotations.results[0].eps_id 6. 查询自定义字段 $event.annotations.results[0].fields.xxx **说明** xxx表示原始日志的结构化字段和内置字段（hostIP、hostName等），日志字段长度最多1KB，超过1KB被截断丢弃。 ● SQL告警支持的变量 1. 图表0的日志组/流名称：$event.annotations.results[0].resource_id **说明** 只支持添加首次创建的日志组/日志流原始名称，不支持添加修改后的日志组/日志流名称。 0代表第一个图表，1代表第二个图表，以此类推。 2. 图表0的查询语句：$event.annotations.results[0].sql 3. 图表0的查询时间：$event.annotations.results[0].time 4. 图表0的查询URL：$event.annotations.results[0].url 5. 图表0的查询日志：$event.annotations.results[0].raw_results	

参数名称	说明	校验规则	样例
		6. 图表0的日志流的企业项目ID：$event.annotations.results[0].eps_id **复制模板：** • keywords_template • sql_template • sql模板 • 关键词模板 • 自定义模板（用户通过添加变量创建的消息模板）	

- 用户自定义创建消息模板
- 复制模板创建消息模板

说明

- 邮件内容支持html标签和消息预览。
- 企业微信、钉钉、飞书支持markdown语法和消息预览。
- 针对AOM和LTS，最多可以创建100（包含）条消息模板，如果消息模板数量已达上限100个时，请删除不需要的消息模板后重新创建。

步骤3 配置完成后，单击“确定”。

----结束

编辑消息模板

步骤1 在消息模板列表中，单击消息模板名称行后的“修改”，根据表8-4进行修改，其中“模板名称”不可修改。

说明

内置消息模板不支持编辑。

步骤2 编辑完成后，单击“确认”。

----结束

复制消息模板

步骤1 在消息模板列表中，单击消息模板名称行后的“复制”，须修改消息模板的模板名称。

步骤2 完成后，单击“确认”。

----结束

删除消息模板

删除消息模板

步骤1 在消息模板列表中，单击消息模板名称行后的“删除”。

说明

内置消息模板不支持删除。

步骤2 在弹出的对话框中，单击“确认”删除该消息模板。

----结束

批量删除消息模板

步骤1 在消息模板列表中，勾选待删除的消息模板，单击列表左上方“批量删除”。

步骤2 在弹出的删除消息模板页面，单击“确定”，删除所勾选的消息模板。

----结束

8.3.2 创建告警行动规则

LTS提供告警行动规则定制功能，您可以通过创建告警行动规则关联SMN主题与消息模板，创建消息模板时，自定义通知消息配置。

前提条件

- 已创建一个主题。详细操作请参考**创建主题**。
- 已设置主题策略。详细操作请参考**设置主题策略**。
- 已为主题添加相关的订阅者，即通知的接收人（例如：邮件或短信）。详细操作请参考**订阅主题**。

注意事项

您最多可创建1000个告警行动规则，如果告警行动规则数量已达上限1000时，请删除不需要的行动规则。

创建告警行动规则

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“日志告警”。

步骤3 单击“告警行动规则”页签。

步骤4 在告警行动规则页签，单击“创建”。设置行动规则名称、行动规则配置等信息。

图 8-1 创建告警行动规则

表 8-5 告警行动规则参数说明

参数名称	说明
行动规则名称	只能由数字、字母、中文、下划线、中划线组成，且不能以下划线、中划线开头结尾，长度为1到64个字符。
企业项目	选择已创建的企业项目。 如果当前账号未开通企业项目则不显示该参数。
描述	行动规则的描述。
主题	SMN主题，请从下拉列表中选择。 若没有合适的主题，请单击主题选择栏下方“创建主题”，在SMN界面创建。
消息模板	通知消息的模板，请从下拉列表中选择。 若没有合适的消息模板，请单击消息模板选择栏右侧“创建消息模板”，新建消息模板，操作详见**创建消息模板**。

步骤5 设置完成后，单击“确定”。

----结束

更多操作

告警行动规则创建完成后，您还可以执行**表8-6**中的相关操作。

表 8-6 相关操作

操作	说明
修改告警行动规则	单击“操作”列的“修改”。
导出告警行动规则	选中单个或多个告警行动规则，单击“导出”，若没有选中告警行动规则，则导出全部告警行动规则。
删除告警行动规则	• 删除单条规则：单击对应规则“操作”列的“删除”，随后在提示页面单击“确定”即可删除。 • 删除单条或多条规则：勾选对应规则前的复选框，单击“批量删除”，随后在提示页面单击“确定”即可删除。 **说明** 删除告警行动规则前需要先删除该行动规则绑定的告警规则。
搜索告警行动规则	在右上角的搜索框中输入规则名称关键字，单击  后显示匹配对象。

8.3.3 模板内容定制

邮件内容支持html标签和消息预览，企业微信、钉钉、飞书支持markdown语法和消息预览。由于每种语法不同，云日志服务支持您在配置消息模板时，根据实际情况定制通知内容信息。

- 邮件

  邮件渠道的内容支持HTML标签。例如：

  使用
换行。

  使用查看详情添加链接。您可以单击该链接查看触发告警的详细信息。使用<span style="color: red">${event_severity} 添加颜色。

- 企业微信

  企业微信渠道的内容支持Markdown语法，。目前支持的元素参考如下：

  标题

  ```
  # 一级标题
  ## 二级标题
  ### 三级标题
  #### 四级标题
  ##### 五级标题
  ###### 六级标题
  ```

  文字加粗

  ```
  **告警**
  ```

  链接

  ```
  [查询url](http://example.com)
  ```

  字体颜色

  ```
  <font color="#FF0000">${event_severity}</font>
  ```

- 钉钉

  钉钉渠道的内容支持Markdown语法，。目前支持的元素参考如下：

标题

```
# 一级标题
## 二级标题
### 三级标题
#### 四级标题
##### 五级标题
###### 六级标题
```

文字加粗

```
**告警**
```

链接

```
[查询url](http://example.com)
```

字体颜色

```
<font color="#FF0000">${event_severity}</font>
```

- 飞书

  飞书渠道的内容支持Markdown语法，。目前支持的元素参考如下：

  加粗

  ```
  **粗体**
  ```

  超链接

  ```
  <a>https://open.feishu.cn</a>
  ```

8.4 查看 LTS 告警列表

云日志服务支持对日志数据进行监控，通过配置关键词告警规则或sql告警规则，定时查询日志数据，当设置的匹配条数或条件表达式满足时，将告警进行上报，用户可以在LTS控制台查看告警。

前提条件

已创建告警规则。

查看告警

步骤1 在云日志服务管理控制台，单击“日志告警”。

步骤2 默认显示“告警列表”页面，在该页面默认显示30分钟（相对）的所有告警列表及其趋势图。

步骤3 输入查询条件后进行搜索，页面会展示该条件下的所有告警信息及这些告警的趋势图，具体查询条件如下：

- 在页面上方搜索框中可根据日志组、日志流和告警级别进行搜索。
- 设置时间范围，默认时间范围为30分钟（相对）。

  时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。

说明

- 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。
- 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。
- 自定义：表示查询指定时间范围的日志数据。

步骤4 设置搜索条件后，单击，查找在已设时间范围内满足搜索条件的告警。

步骤5 查询的告警默认显示在“活动告警”页签下，将鼠标放在目标告警所在行中的“告警详情”可查看告警详情。单击告警列表中对应的“名称”，界面右侧弹出该告警的详细信息。

告警故障已经解除时，可单击列表中告警所在行后的删除按钮对该告警进行清除，被执行清除操作后的告警将会显示在“历史告警”页签。

针对已设置好的搜索条件，告警列表默认需要手动刷新，如需设置自动刷新可单击告警界面右上角 ，在弹出的下拉列表中选择“30秒自动刷新”、“1分钟自动刷新”或“5分钟自动刷新”，若在设置自动刷新后需要手动刷新，也可在下拉列表重新选择“手动刷新”。

----结束

9 日志转储

9.1 日志转储概述

主机和云服务的日志数据上报至云日志服务后，默认存储时间为30天，您在创建日志组时，可以对日志存储进行设置（1-365天）。超出存储时间的日志数据将会被自动删除，对于需要长期存储的日志数据（日志持久化），云日志服务提供转储功能，可以将日志转储至其他云服务中进行长期保存。

说明

日志转储功能只能拷贝已有日志，不会删除日志。云日志服务LTS根据用户配置的日志存储时间定时清理日志文件，不会影响转储后的日志。

- 当前LTS支持转储至以下云服务，根据您的业务场景进行日志转储。
  - **日志转储至OBS**

    对象存储服务（Object Storage Service，OBS）提供海量、安全、高可靠、低成本的数据存储能力，可供用户存储任意类型和大小的数据。
  - **日志转储至DIS**

    数据接入服务（Data Ingestion Service,简称DIS）提供日志长期存储能力和丰富的大数据分析能力。
  - **日志转储至DMS**

    DMS通过分布式消息服务API处理日志。
  - **日志转储至DWS**

    数据仓库服务（Data Warehouse Service，简称DWS）是完全托管的企业级云上数据仓库服务，具备免运维、在线扩展、高效的多源数据加载能力，兼容PostgreSQL生态。助力企业经济高效地对海量数据进行在线分析，实现数据快速变现。
  - **日志转储至DLI**

    数据湖探索（Data Lake Insight，简称DLI）是完全兼容Apache Spark、Apache Flink、openLooKeng（基于Apache Presto）生态，提供一站式的流处理、批处理、交互式分析的Serverless融合处理分析服务。用户不需要管理任何服务器，即开即用。支持标准SQL/Spark SQL/Flink SQL，支持多种接入方式，并兼容主流数据格式。

- 当前LTS支持跨租户转储，可以通过**创建委托**将日志转储的能力共享给其他账号，让其他账号代为转储该账号下的日志数据。委托时需要对被委托账号赋予可以获取日志组和日志流的权限，例如LTS FullAccess。

9.2 日志转储至 OBS

对象存储服务 OBS提供日志存储功能；您可以将日志转储至OBS，并在OBS控制台下载日志文件。支持将日志将周期性或一次性的转储至对象存储服务（OBS）中长期保存。

说明

- 创建日志转储时，除需拥有LTS使用权限外，还需要拥有OBS Administrator权限。
- 云日志服务配置的日志转储是将最新产生的日志转储到OBS桶中，不会对历史日志进行转储。

前提条件

- 日志已接入LTS。
- 已创建OBS桶。

说明

OBS存储独立收费，收费详情请参见：**华为云定价**。

创建日志转储（周期性）

步骤1 在云日志服务控制台，左侧导航栏中，单击“日志转储”。

步骤2 在“日志转储”页面右上角，单击“配置转储”。

步骤3 在“配置转储”页面，设置转储日志相关参数。

表 9-1 配置转储参数说明

参数名称	说明	样例
日志源	• 当前账号：对用户所在账号下所产生的日志进行转储。 • 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中**创建委托**。	当前账号
转储方式	日志源选择当前账号时，支持选择周期性转储或一次性转储。 • 周期性转储：日志将周期性的转储至对象存储服务（OBS）中长期保存。 • 一次性转储：日志将一次性的转储至对象存储服务（OBS）中长期保存。	周期性转储
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-

参数名称	说明	样例
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	默认开启转储。	开启
转储对象	选择转储的云服务。	OBS
日志组名称	选择已创建的日志组。	-
企业项目	选择已创建的企业项目。 ● 如果当前账号未开通企业项目则不显示该参数。 ● 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	-
日志流名称	选择已创建的日志流。 **说明** 已配置过OBS转储的日志流不能重复配置。	-
OBS桶	● 选择已创建的OBS桶。 – 如果没有可选择的OBS桶，单击“查看OBS”，进入对象存储服务管理控制台，创建OBS桶。 – 如果OBS桶为加密桶，则需要选择“密钥名称”，并勾选下方的“我同意在KMS创建授权给LTS账号，对转储日志加解密”。 ● LTS目前仅支持单AZ存储策略、标准存储类别的OBS桶。 **说明** 首次配置一次性转储到未授权的OBS桶中时，LTS服务会授权给OBS桶ACL规则，授权生效需要15分钟，如果您第一次配置一次性转储后失败，请15分钟后重试。请谨慎修改桶策略，防止转储失败。	-
密钥名称	对于加密的OBS桶，选择密钥名称。如果没有可选择的密钥，单击“创建密钥并授权”，进入数据加密控制台，创建密钥。	-

参数名称	说明	样例
自定义转储路径	• 开启：将日志转储至自定义路径中，用于区分不同日志流之间的转储日志文件。 格式为：/LogTanks/RegionName/*自定义转储路径*。自定义转储路径默认为lts/%Y/%m/%d，其中%Y代表年，%m代表月，%d代表日，格式需要符合如下规范： – “/LogTanks/RegionName”为系统默认路径，不可以修改。 – 名称只能由英文字母、数字及特殊字符“&”“$”“@”“;”“:”“,”“=”“+”“?”“-”“.”“_”“/”和“%”组成，且“%”后只可跟Y（年）、m（月）、d（日）、H（时）、M（分），在%Y、%m、%d、%H和%M前后可以添加任意长度字符，并且可对其先后顺序进行调换。 – 自定义转储路径名称不允许为空，长度限制为1~128个字符。 – 新增%GroupName代表日志组名称，%StreamName代表日志流名称。 示例： 1. 输入LTS-test/%Y/%m/%done/%H/%m，则日志转储路径为*：LogTanks/RegionName/LTS-test/Y/m/done/H/M/日志文件名称*。 2. 输入LTS-test/%d/%H/%m/%Y，则日志转储路径为：*LogTanks/RegionName/LTS-test/d/H/m/Y/日志文件名称*。 • 不开启：将日志转储至系统默认路径中。系统默认路径为：LogTanks/RegionName/*2019/01/01/日志组/日志流/日志文件名称*。	LTS-test/%Y/%m/%done/%H/%M
日志文件前缀	转储至OBS桶中的日志文件前缀。 日志文件前缀需符合如下规范： • 名称长度限制为0~64个字符。 • 名称只能由英文大小写字母、数字、中划线“-”、下划线“_”和小数点“.”组成。 示例：输入LTS-log，则日志文件名称为：LTS-log_日志文件名称。	LTS-log

参数名称	说明	样例

转储格式	用于配置日志的转储格式，可选择“原始日志格式”、“Json格式”和ORC。
	说明
	ORC转储格式（包括ORC字段和无效字段填充）仅支持华东-上海一的白名单用户使用，其他局点暂不支持该功能。
	● 原始日志格式示例：
	云日志服务控制台展示的日志内容的格式为原始日志格式。
	`Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)`
	● JSON格式示例：
	`{"host_name":"ecs-bd70","ip":"192.168.0.54","line_no":249,"message":"Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\n","path":"/var/log/syslog","time":1569825602303}`
		Json

ORC字段	转储格式选择ORC时需要设置ORC字段。 支持自动配置字段或者单击添加，在下拉框选择或输入键值、选择类型。 ● 键：日志字段名称，同一个字段名在ORC字段中只能配置一次，不支持多次使用。 ● 类型：string、boolean、int、long、float、double。 日志转储过程中，会将云日志服务中的日志字段由string类型转换为ORC目标类型。如果转换到非string类型失败，int/long/float/double类型会置为默认值0。	-
无效字段填充	转储格式选择ORC时支持设置无效字段填充。 若日志中没有值与上方配置的键值对应，则会用无效字段填充。	-
转储周期	日志自动转储至OBS桶的时间间隔，支持2分钟、5分钟、30分钟、1小时、3小时、6小时、12小时。	3小时
文件名时区	日志自动转储至OBS桶时，按照UTC时间生成转储目录及文件名称。	(UTC)协调世界时间

参数名称	说明	样例
是否投递tag	如主机日志，转储时会增加采集器收集的tag字段。 ● 不开启：不会投递tag。 ● 开启：默认的投递tag有：主机信息（hostIP、hostId、hostName、pathFile、collectTime）；kubernetes信息（clusterName、clusterId、nameSpace、podName、appName、containerName）。可选择公共tag有：regionName、projectId、logStreamName、logGroupName。 **说明** 当开启投递tag后，转储格式必须是JSON格式。 ● 转储标签：开启后，会将日志流标签添加至转储内容。	开启
压缩格式	支持不压缩、压缩gzip、压缩zip、压缩snappy。 **说明** 压缩格式支持白名单用户申请**提交工单**开通。	gzip

步骤4 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

步骤5 单击“转储对象”列的OBS桶名称，可以跳转至OBS控制台，查看转储的日志文件。

转储到OBS后的日志，支持从OBS下载到本地进行查看。

说明

转储至OBS的日志支持下载的格式：原始日志、JSON格式。

----结束

创建一次性日志转储

步骤1 在云日志服务管理控制台，左侧导航栏中，单击“日志转储”。

步骤2 在“日志转储”页面右上角，单击“配置转储”。

步骤3 在“配置转储”页面，设置转储日志相关参数。

表 9-2 配置转储参数说明

参数名称	说明	样例
转储方式	● 周期性转储：日志将周期性的转储至对象存储服务（OBS）中长期保存。 ● 一次性转储：日志将一次性的转储至对象存储服务（OBS）中长期保存。	一次性转储

参数名称	说明	样例
转储对象	选择转储的云服务。	OBS
日志组名称	选择已创建的日志组。	-
企业项目	选择已创建的企业项目。 ● 如果当前账号未开通企业项目则不显示该参数。 ● 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	-
日志流名称	选择已创建的日志流。 **说明** 已配置过OBS转储的日志流不能重复配置。	-
过滤条件	默认关键词过滤，在输入框填写需要过滤的关键词。	-
转储时间范围	时间范围有三种方式，分别是相对时间、整点时间和自定义。您可以根据自己的实际需求，选择时间范围。 ● 相对时间：表示查询距离当前时间1分钟、5分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置相对时间1小时，表示查询18:20:31~19:20:31的日志数据。 ● 整点时间：表示查询最近整点1分钟、15分钟等时间区间的日志数据。例如当前时间为19:20:31，设置整点时间1小时，表示查询18:00:00~19:00:00的日志数据。 ● 自定义：表示查询指定时间范围的日志数据	-
日志总条数	日志总条数。	-
转储文件个数	单次一次性转储日志条数上限2000万条，转储文件个数上限200个。	-

参数名称	说明	样例
OBS桶	● 选择已创建的OBS桶。 如果没有可选择的OBS桶，单击“查看OBS”，进入对象存储服务管理控制台，创建OBS桶。 ● LTS目前仅支持存储类别为“标准存储”的OBS桶。 **说明** 首次配置一次性转储到未授权的OBS桶中时，LTS服务会授权给OBS桶ACL规则，授权生效需要15分钟，如果您第一次配置一次性转储后失败，请15分钟后重试。请谨慎修改桶策略，防止转储失败。	-
所属桶目录	所属OBS桶目录。	-
转储文件名称	自定义转储文件名称，只能由英文字母、数字、中划线、下划线、小数点组成。	-

转储格式	用于配置日志的转储格式，可选择原始日志格式、Json格式、CSV格式。
	● 原始日志格式示例：
	云日志服务控制台展示的日志内容的格式为原始日志格式。
	`Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)`
	● JSON格式示例：
	`{"host_name":"ecs-bd70","ip":"192.168.0.54","line_no":249,"message":"Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\n","path":"/var/log/syslog","time":1569825602303}`
	● CSV格式：以表格的形式展示日志内容。
		Json

步骤4 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。

步骤5 单击“转储对象”列的OBS桶名称，可以跳转至OBS控制台，查看转储的日志文件。

步骤6 转储到OBS后的日志，支持从OBS下载到本地进行查看。

----结束

修改日志转储

1. 在日志转储列表中，单击待修改配置转储任务所在行的“修改”，弹出“修改转储”对话框，进行修改。
2. 修改完成后，单击“确定”。

查看转储详情

1. 在日志转储列表中，单击待查看配置转储任务所在行的“更多>详情”。
2. 在弹出的“转储详情”页面中，可查看日志转储详情。

删除转储任务

如果日志不再需要转储，可以删除转储任务。

说明

- 转储任务一旦删除将不再对日志进行转储，请谨慎操作。
- 删除转储任务后，之前已经转储日志将会继续保存在OBS。
- 创建转储任务时，选中的OBS桶会将读写策略授权给云日志服务。当多个转储任务使用同一OBS桶时，如您需要删除转储任务，请按如下操作：
  - 如果仅使用该OBS桶创建了一个转储任务，删除该转储任务时，请在对象存储服务（Object Storage Service，OBS）中，**“访问权限控制”>“桶ACLs”**里删除特定用户的桶访问权限。
  - 如果使用该OBS桶创建了多个转储任务，请勿删除桶访问权限，否则会导致转储失败。

1. 在日志转储列表中，单击待删除的日志组所在行的“删除”，弹出“删除”对话框。
2. 单击“确定”，删除转储任务。

查看转储状态

日志转储任务的转储状态共分为正常、异常、关闭三种状态。

- 正常：日志转储任务正常进行。
- 异常：日志转储任务异常，可能是如下原因导致：
  - OBS桶策略异常，请您在对象存储服务中设置访问控制策略。
  - OBS加密桶的密钥被删除或被取消授权，请您确保授权密钥的合法性。
- 关闭：日志转储任务停止。

9.3 日志转储至 DIS

DIS提供丰富的大数据分析能力，可以将大量日志文件传输到云端做备份，进行离线分析、存储查询及机器学习，还能用于数据丢失或异常后的恢复和故障分析。同时大量小文本文件可合并转储为大文件，提高数据处理性能。您可以根据业务场景选择是否使用DIS进行日志转储。

说明

- 建议您优先使用**转储至DMS**。
- 目前此功能仅支持华北-乌兰察布二零一、华北-乌兰察布二零二、华北-北京四、华北-北京一、华东-上海二、华南-广州、中国-香港、亚太-新加坡、华东-上海一局点，其他局点需要给DIS服务提交工单申请开通才能使用。详细操作请参考**提交工单**。

前提条件

- 日志已接入LTS。
- 已购买DIS。

说明

DIS存储独立收费，收费详情请参见：**华为云定价**。

日志转储至 DIS

步骤1 在云日志服务管理控制台，左侧导航栏中，单击“日志转储”。

步骤2 在“日志转储”页面右上角，单击“配置转储”。

步骤3 在“配置转储”页面，设置转储日志相关参数。

表 9-3 配置转储参数说明

参数名称	说明	样例
日志源	• 当前账号：对用户所在账号下所产生的日志进行转储。 • 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中**创建委托**。	当前账号
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	选择开启转储。	开启
转储对象	选择转储的云服务。	DIS
日志组名称	选择已创建的日志组。	-
企业项目	选择已创建的企业项目。 • 如果当前账号未开通企业项目则不显示该参数。 • 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	-
日志流名称	选择已创建的日志流。 **说明** 已配置过DIS转储的日志流不能重复配置。	-
通道名称	选择已创建的DIS通道。如果没有可选择的通道，单击“查看DIS通道”，进入数据接入服务管理控制台，创建接入通道。	-

参数名称	说明	样例

转储格式	用于配置日志的转储格式，可选择“原始日志格式”和“JSON格式”。
	● 原始日志格式示例：
	云日志服务控制台展示的日志内容的格式为原始日志格式。
	`Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)`
	● JSON格式示例：
	`{"host_name":"ecs-bd70","ip":"192.168.0.54","line_no":249,"message":"Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\n","path":"/var/log/syslog","time":1569825602303}`
		JSON

转储周期	日志将实时转储至DIS通道中。	实时
是否投递tag	如主机日志，转储时会增加采集器收集的tag字段。 ● 不开启：不会投递tag。 ● 开启：默认的投递tag有：主机信息（hostIP、hostId、hostName、pathFile、collectTime）；kubernetes信息（clusterName、clusterId、nameSpace、podName、appName、containerName）。可选择公共tag有：regionName、projectId、logStreamName、logGroupName。 **说明** 当开启投递tag后，转储格式必须是JSON格式。 ● 转储标签：开启后，会将日志流标签添加至转储内容。	开启

步骤4 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。当选择对其他账号日志进行转储时，被委托人的转储界面，日志组和流属于委托人，前端点击日志组、流连接时，需要通过委托跳转到委托人的日志组、流界面。

步骤5 单击“转储对象”列的DIS通道名称，可以跳转至DIS控制台，查看转储的日志文件。

转储后的日志，支持下载到本地进行查看。

说明

当删除该转储任务时，请在数据接入服务（Data Ingestion Service，DIS）中，单击“通道管理”，选择该DIS实例进入实例详情页面。在授权管理中，删除上传权限。

----结束

9.4 日志转储至 DMS

分布式消息服务 DMS提供日志实时处理管道，您可以通过分布式消息服务API实时消费处理日志。

说明

目前此功能仅支持白名单用户提交工单申请使用。详细操作请参考**提交工单**。

前提条件

- 日志已接入LTS。
- 已购买DMS。

  说明

  DMS存储独立收费，收费详情请参见：**华为云定价**。
- 在注册DMS Kafka实例前，需在安全组中，开放**入方向规则**198.19.128.0/17和9011端口。

日志转储至 DMS

1. 在云日志服务管理控制台，左侧导航栏中，单击“日志转储”。
2. 在“日志转储”页面右上角，单击“配置转储”。
3. 在“配置转储”页面，设置转储日志相关参数。

表 9-4 配置转储参数说明

参数名称	说明	样例
日志源	• 当前账号：对用户所在账号下所产生的日志进行转储。 • 其他账号：对委托人账号下所产生的日志进行转储，如需转储其他账号日志，需该账号使用者在IAM中**创建委托**。	当前账号
委托名称	当转储其他账号时，需填写委托人在IAM中创建的委托名称。	-
委托人账号名称	当转储其他账号时，需填写委托人的账号名称。	-
是否开启转储	默认开启转储。	开启
转储对象	选择转储的云服务。	DMS
日志组名称	选择已创建的日志组。	-

参数名称	说明	样例
企业项目	选择已创建的企业项目。 ● 如果当前账号未开通企业项目则不显示该参数。 ● 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	-
日志流名称	选择已创建的日志流。 **说明** 已配置过DMS转储的日志流不能重复配置。	-
Kafka实例	选择Kafka实例。如果没有可选择的实例，单击“查看Kafka实例”，进入分布式消息管理控制台，创建Kafka专享版。 如果Kafka实例已注册（如果未注册，请注册Kafka示例，操作指导请参见：**注册Kafka实例**），可以选择修改Kafka实例。 **说明** 创建Kafka实例时，设置实例的访问方式：内网访问开启密文接入，“kafka安全协议”选择“SASL_SSL”，设置用户名和密码。同时开启“SASL PLAIN机制”。详细操作请参考**购买实例**。	-
Topic	选择Kafka实例的topic，如果没有可选择的topic，进入分布式消息管理控制台，创建专享版Kafka的topic	topic-01

转储格式	用于配置日志的转储格式，可选择“原始日志格式”和“JSON格式”。
	● 原始日志格式示例：
	云日志服务控制台展示的日志内容的格式为原始日志格式。
	`Sep 30 07:30:01 ecs-bd70 CRON[3459]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)`
	● JSON格式示例：
	`{"host_name":"ecs-bd70","ip":"192.168.0.54","line_no":249,"message":"Sep 30 14:40:01 ecs-bd70 CRON[4363]: (root) CMD (/opt/oss/servicemgr/ICAgent/bin/manual/mstart.sh > /dev/null 2>&1)\n","path":"/var/log/syslog","time":1569825602303}`
		RAW

转储周期	日志将实时转储至Kafka实例中。	实时

参数名称	说明	样例
用户日志字段	当转储格式选择JSON时，需要设置该参数。 是否开启转储用户日志字段。 选择转储所有字段后将转储日志下所有的字段，选择自定义转储字段后将手动配置用户日志字段。	-
LTS内置字段和用户自定义Tag	当转储格式选择JSON时，需要设置该参数。 是否开启LTS内置字段和用户自定义Tag。 选择转储所有字段后将转储日志下所有的内置字段和用户自定义字段，选择自定义转储字段后将手动配置LTS内置字段和自定义字段。	-
日志流标签字段	当转储格式选择JSON时，需要设置该参数。 是否开启日志流标签字段。开启后，转储类型支持转储所有字段或自定义转储字段。 选择转储所有字段后将转储日志下所有的日志流标签字段，选择自定义转储字段后将手动配置日志流标签字段。如何设置标签请参考**管理日志流**。	-
更多配置	当转储格式选择JSON时，需要设置该参数。 若日志中无值与上方配置的键值对应，则会用无效字段填充。	-

4. 单击“确定”，完成配置。当转储任务状态为“正常”时，表示转储任务创建成功。当选择对其他账号日志进行转储时，被委托人的转储界面，日志组和流属于委托人，前端点击日志组、流连接时，需要通过委托跳转到委托人的日志组、流界面。
5. 单击“转储对象”列的名称，可跳转到专享版Kafka实例的基本信息页面。

注册 Kafka 实例

1. 如果选择Kafka实例未注册，单击注册，跳转到注册Kafka实例页面。
2. 注册Kafka实例相关参数说明。

参数名称	说明	样例
Kafka实例	dms实例名称。	Kafka-01
打通DMS网络	打通Kafka实例和LTS服务的网络，用户LTS服务通过该网络发送转储数据。	-
用户名	如果Kafka实例开启了sasl认证，需要输入sasl认证的用户名。	DMS
密码	如果Kafka实例开启了sasl认证，需要输入sasl认证的密码。	-

3. 单击“确认”，完成注册Kafka实例。

9.5 日志转储至 DWS

数据仓库服务GaussDB(DWS) 是一种基于华为云基础架构和平台的在线数据处理数据库，提供即开即用、可扩展且完全托管的分析型数据库服务。转储至数据仓库服务GaussDB(DWS) ，可以将日志中的结构化字段转储到DWS数据库表中，您可以根据业务场景选择是否使用DWS进行日志转储。

说明

目前此功能仅支持华北-北京四、华东-上海一、华南-广州、亚太-新加坡局点，其他局点需要提交工单申请使用。详细操作请参考**提交工单**。

前提条件

- 日志已接入云日志服务（ LTS ）。
- 日志流已配置结构化规则。
- 已购**独享型且规格为网络型的弹性负载均衡（ ELB ）实例**。
- 已**创建DWS集群**，并且绑定弹性负载均衡（ ELB ）。

说明

暂不支持DWS集群跨VPC绑定弹性负载均衡（ ELB ）。

日志转储至 DWS

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志转储 ”。

步骤2 在“日志转储”页面中，单击右上角“配置转储”。

步骤3 在“配置转储”页面中，选择转储对象“DWS集群”，并配置各参数信息。

表 9-5 配置转储参数说明

参数名称	说明	样例
是否开启转储	选择是否开启转储。	开启
转储对象	选择转储的云服务。	DWS集群
日志组名称	选择已创建的日志组。	-

参数名称	说明	样例
企业项目	选择已创建的企业项目。 ● 如果当前账号未开通企业项目则不显示该参数。 ● 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	default
日志流名称	选择已创建的日志流。 **说明** 已配置过DWS集群转储的日志流不能重复配置。	-
集群名称	已创建的集群名称。	test
数据库名称	集群的数据库名称。有两种数据库，分别是“gaussdb”和“postgres”。默认集群数据库为“gaussdb”。	gaussdb
用户名	数据库的管理员用户名。	lts-test
密码	数据库的管理员密码。	-
schema名称	数据库对象的集合名称。	-
表名	schema中的表名称。	-

参数名称	说明	样例
字段映射	将内置字段以及日志中配置的结构化字段和类型，映射到数据库字段。 **说明** 内置字段有13个，分别是hostIP、 hostId、hostName、pathFile、collectTime、clusterName、clusterId、 podName、containerName、regionName、projectId、logGroupName和logStreamName 当结构化字段类型和数据库表字段类型一致时，支持将日志的结构化字段转储至数据仓库服务GaussDB(DWS)，否则转储无效。 • 可以通过结构化字段和表字段的下拉框，选择您需要转储的字段。 • 可以通过操作列下的删除操作，选择您需要转储的字段。 • 可以通过单击添加添加，选择您需要转储的字段。	hostIP

步骤4 完成后单击“确定”。

----结束

9.6 日志转储至 DLI

数据湖探索（Data Lake Insight，简称DLI）是完全兼容Apache Spark、Apache Flink、openLooKeng（基于Apache Presto）生态，提供一站式的流处理、批处理、交互式分析的Serverless融合处理分析服务。用户不需要管理任何服务器，即开即用。支持标准SQL/Spark SQL/Flink SQL，支持多种接入方式，并兼容主流数据格式。数据无需复杂的抽取、转换、加载，使用SQL或程序就可以对云上CloudTable、RDS、DWS、CSS、OBS、ECS自建数据库以及线下数据库的异构数据进行探索。

基于转储DLI功能，您可以轻松将LTS中的日志按照字段映射关系转储到DLI数据库表中，进行后续的大数据分析工作。

说明

目前此功能仅在华北-北京四、华南-广州、华东-上海一局点支持白名单用户提交工单申请使用，详细操作请参考**提交工单**，其他局点暂不支持该功能。

前提条件

- 日志已接入云日志服务（LTS）。
- 日志流已配置结构化规则。
- 已在DLI中创建数据库和表，创建数据库表时数据位置选择OBS，数据格式选择JSON。详细操作请参见**创建数据库和表**。
- 已在DLI中创建队列，该队列将用来将数据导入DLI的表中。详细操作请参见**创建队列**。

日志转储至 DLI

步骤1 登录云日志服务控制台，在左侧导航栏中选择“日志转储 ”。

步骤2 在“日志转储”页面中，单击右上角“配置转储”。

步骤3 在“配置转储”页面中，选择转储对象“DLI集群”，并配置各参数信息。

表 9-6 配置转储参数说明

参数名称	说明	样例
是否开启转储	选择是否开启转储。	开启
转储对象	选择转储的云服务。	DLI集群
日志组名称	选择已创建的日志组。	-
企业项目	选择已创建的企业项目。 ● 如果当前账号未开通企业项目则不显示该参数。 ● 如果当前账号已开通企业项目，则存在以下情况： – 当转储当前账号日志时，下拉框显示当前账号的全部企业项目。 – 当转储其他账号日志时，若委托账号未开通企业项目，则默认显示“default”。 – 当转储其他账号日志时，若委托账号已开通企业项目，则显示委托账号的全部企业项目。	default
日志流名称	选择已创建的日志流。 **说明** 已配置过DLI集群转储的日志流不能重复配置。	-
DLI-数据库	需要转储的目标DLI数据库名称。更多信息请参见**DLI库表管理**。	test
DLI-数据表	需要转储的目标DLI数据库表名称。更多信息请参见**DLI库表管理**。	-

参数名称	说明	样例
表普通列映射	将内置字段以及日志中配置的结构化字段和类型，映射到数据库表字段。 **说明** 内置字段有13个，分别是hostIP、 hostId、hostName、pathFile、collectTime、clusterName、clusterId、podName、containerName、regionName、projectId、logGroupName和logStreamName ● 可以通过结构化字段和表字段的下拉框，选择您需要转储的字段。 ● 可以通过操作列下的删除操作，选择您需要转储的字段。 ● 可以通过单击添加添加，选择您需要转储的字段。 ● 表普通列映射和表分区列映射添加表字段的总数为创建表的总列数。	-
表分区列映射	存储时根据设置的字段值进行分区。将内置字段以及日志中配置的结构化字段和类型，映射到数据库表字段。 **说明** 内置字段有13个，分别是hostIP、 hostId、hostName、pathFile、collectTime、clusterName、clusterId、podName、containerName、regionName、projectId、logGroupName和logStreamName ● 可以通过结构化字段和表字段的下拉框，选择您需要转储的字段。 ● 可以通过操作列下的删除操作，选择您需要转储的字段。 ● 可以通过单击添加添加，选择您需要转储的字段。只能添加一个表字段。	-

步骤4 完成后单击“确定”。

----结束

10 日志消费与加工

10.1 DSL 加工（邀测）

10.1.1 DSL 加工概述

DSL（Domain Specific Language）加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、脱敏、过滤等加工任务。

说明

DSL加工的功能在邀测中，支持华北-北京四、华东-上海一、华南-广州局点，仅针对用户内测使用，后续将全网开放，敬请期待！

背景信息

用户对采集到LTS的日志流有二次加工的诉求，目前使用函数加工有以下缺点：

1. 用户有多条日志流需要转换成不同的日志流结构，一个函数只能转换一个日志流，使用不方便，有新增日志结构的时候就需要新增函数，功能繁琐，使用不够灵活。
2. 函数加工需要使用函数工作流服务，且还需要独立收费，长期使用增加成本。

应用场景

- 提取结构化的数据，方便后续检索分析，生成仪表盘等。
- 日志减肥瘦身，节约后续使用成本。丢弃不需要的日志数据，节约存储成本、流量成本。
- 敏感数据脱敏，例如：将用户的身份证、手机号码脱敏。
- 日志分类投递，例如：按照日志级别：ERROR、WARNING、INFO 将日志分类，然后分发到不同的日志主题。

方案介绍

云日志服务支持通过创建DSL加工任务，将用户源日志流数据进行加工输出到目标日志流中，主要操作流程如下：

1. 通过协同消费组对源日志流的数据进行读取。
2. 通过加工规则对读取到的每一条数据进行加工处理。
3. 将加工后的数据写入目标日志流。数据加工完成后，您可以在目标日志流中查看加工后的数据。

功能特性

云日志服务提供数据加工功能，用于数据的规整、富化、脱敏和过滤。

- 数据规整：针对混乱格式的日志进行字段提取、格式转换，获取结构化数据以支持后续的流处理、数据仓库计算。
- 数据富化：对日志（例如订单日志）和维表（例如用户信息表）进行字段连接（JOIN），为日志添加更多维度的信息，用于数据分析。
- 数据脱敏：对数据中包含的密码、手机号、地址等敏感信息进行脱敏。
- 数据过滤：过滤出关键服务的日志，用于重点分析。

10.1.2 创建 DSL 加工任务

DSL（Domain Specific Language）加工是LTS为您提供的一站式日志加工平台，基于领域自定义的脚本语言和200多个内置函数，您可以在LTS控制台实现端到端的日志规整、富化、流转、脱敏、过滤等加工任务。详细加工语法请参考**DSL数据加工语法（邀测）**。

前提条件

- 已成功采集到日志。
- 对源日志内容已完成结构化配置，具体操作请参考**结构化配置**。

创建 DSL 加工任务

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“日志加工>DSL加工”，单击“新建DSL加工任务”。

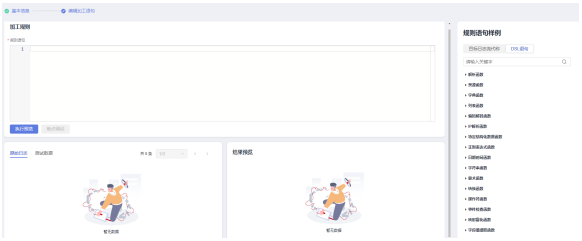
步骤3 在基本信息页面中，完成如下配置后，然后单击“下一步”。

表 10-1 基本信息参数

参数	说明
任务名称	DSL加工任务的名称。长度范围为1~128个字符。
启用状态	默认开启。 **说明** 创建DSL加工任务，需要您授权LTS创建一个云服务委托：LTS

参数	说明
源日志流	• 当前账号 选择**已完成结构化配置的日志组、日志流**，即表示源日志组、日志流中的日志内容通过DSL加工处理后，将存储到目标日志流中。 • 其他账号 填写委托名称、委托人账号名称，选择日志组和日志流。关于委托信息请参考**创建委托**。
目标日志流	单击“添加目标日志流”，填写目标日志流名称（目标日志流在数据加工函数中作为入参时的代称），选择日志组和日志流。
高级配置	对于加工语句中需要使用的密码信息（例如数据库连接密码），日志服务支持使用键值对形式保存在密钥对中。 单击“添加密钥”，填写key值、Value值。

步骤4 在编辑加工语句页面中，参考规则语句样例填写规则语句，单击执行预览。关于加工语法请参考**DSL数据加工语法（邀测）**。



步骤5 结果预览显示正常，单击“确定”。创建成功后在DSL加工页面生成1条任务明细。

说明

支持将所有加工任务导出全部数据到XLSX。

----结束

10.2 DSL 数据加工语法（邀测）

10.2.1 概述

字段提取检查与覆盖模式，介绍字段提取模式mode参数的不同取值以及说明。请参考**DSL加工（邀测）**创建加工任务。

说明

DSL加工的功能在邀测中，支持华北-北京四、华东-上海一、华南-广州局点，仅针对用户内测使用，后续将全网开放，敬请期待！

表 10-2 参数说明

参数值	说明
fill	当目标字段不存在或者值为空时，设置目标字段。
fill-auto	当新值非空，且目标字段不存在或者值为空时，设置目标字段。
add	当目标字段不存在时，设置目标字段。
add-auto	当新值非空，且目标字段不存在时，设置目标字段。
overwrite	总是设置目标字段。
overwrite-auto	当新值非空，设置目标字段。

10.2.2 操作符函数

10.2.2.1 解析函数

本文介绍User-Agent解析函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
ua_parse_device	解析User-Agent中的设备信息。
ua_parse_os	解析User-Agent中的操作系统信息。
ua_parse_agent	解析User-Agent中的浏览器信息。
ua_parse_all	解析User-Agent中所有信息。
url_parse	解析URL的组成部分。
url_parse_qs	解析URL中查询字符串包含的参数。

说明

User-Agent解析函数会剔除解析结果为None的字段，例如解析的设备数据为{'brand': None, 'family': 'Other', 'model': None}，则brand字段和model字段将被剔除，最终的解析结果为{'family': 'Other'}。

ua_parse_device

解析User-Agent中的设备信息。

- **函数格式**


```
ua_parse_device(value)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- **返回结果**

  返回JSON类型的数据集。

- **函数示例**
  - 测试数据

    ```
    {
      "http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
    }
    ```

  - 加工规则

    ```
    e_set("new_column",ua_parse_device(v("http_user_agent")))
    ```

  - 加工结果

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    new_column:{"family":"Mac","brand":"Apple","model":"Mac"}
    ```

ua_parse_os

解析User-Agent中的操作系统信息。

- **函数格式**

  ```
  ua_parse_os(value)
  ```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- **返回结果**

  返回JSON类型的数据集。

- **函数示例**
  - 测试数据

    ```
    {
      "http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
    }
    ```

  - 加工规则

    ```
    e_set("new_column",ua_parse_os(v("http_user_agent")))
    ```

  - 加工结果

    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    new_column:{'family': 'Mac OS X',
                'major': '10',
                'minor': '9',
                'patch': '4'}
    ```


ua_parse_agent

解析User-Agent中的浏览器信息。

- **函数格式**
  ua_parse_agent(value)
- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- **返回结果**
  返回JSON类型的数据集。
- **函数示例**
  - 测试数据
    ```
    {
      "http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
    }
    ```
  - 加工规则
    ```
    e_set("new_column",ua_parse_agent(v("http_user_agent")))
    ```
  - 加工结果
    ```
    http_user_agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    new_column:{'family': 'Chrome', 'major': '192', 'minor': '168', 'patch': '0'
    ```

ua_parse_all

解析User-Agent中的操作系统信息。

- **函数格式**
  ua_parse_all(value)
- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	待解析的User-Agent字符串。

- **返回结果**
  返回JSON类型的数据集。
- **函数示例**
  - 测试数据
    ```
    {
      "http_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36"
    }
    ```
  - 加工规则
    ```
    e_set("new_column",ua_parse_all(v("http_user_agent")))
    ```
  - 加工结果
    ```
    http_user_agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/192.168.0.0 Safari/537.36
    ```

```
new_column: {
 "user_agent": {
   "family": "Chrome","
    major""major": "192",
    "minor": "168",
    "patch": "0"
   },
 "os": {
    "family": "Mac OS X",
    "major": "10",
    "minor": "9",
    "patch": "4"
 },
 "device": {
   "family": "Mac",
    "brand": "Apple",
    "model": "Mac"
 }
}
```

url_parse

- **函数格式**

```
url_parse(url, scheme="", allow_fragments=true)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	待解析的URL。
scheme	String	否	网络协议，默认为空字符。仅在URL中未指定网络协议时，返回结果中的scheme字段才会使用此处设置的值。
allow_fragments	Boolean	否	是否解析URL中的fragment部分。 ● true（默认值）：解析URL中的fragment部分，返回结果中的fragment字段为具体值。 ● false：不解析URL中的fragment部分，返回结果中的fragment字段为空字符串。

- **返回结果**

返回解析后的JSON数据，具体参数说明如下表所示。

字段	说明
scheme	网络协议
netloc	网络位置
path	分层路径标识
query	查询组件
fragment	片段标识符

- **函数示例**

  a. 示例1：使用默认参数，返回URL的各个组成部分。

     - 测试数据

       ```
       {
          "content":"https://username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib"
       }
       ```

     - 加工规则

       ```
       e_set("url",url_parse(v("content")))
       ```

     - 加工结果

       ```
       content:https://username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib
       url:{
              "scheme": "https",
               "netloc": "username:username@example.com:8083",
               "path": "/hello/asdah/",
               "params":  "type=docx",
               "query": "filename=python3.docx",
               "fragment": "urllib",
           }
       ```

  b. 示例2：设置allow_fragments为false，返回结果中的fragment参数值为空。

     - 测试数据

       ```
       {
          "content":"https://username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib"
       }
       ```

     - 加工规则

       ```
       e_set("url",url_parse(v("content"),allow_fragments=false))
       ```

     - 加工结果

       ```
       content:https://username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib
       url:{
                    "scheme": "https",
              "netloc": "username:username@example.com:8083",
              "path": "/hello/asdah/",
               "params": "type=docx",
               "query": "filename=python3.docx",
              "fragment": "",
           }
       ```

  c. 示例3：设置scheme为https，allow_fragments为false，返回结果中scheme参数值为https，fragment参数值为空。

     - 测试数据

       ```
       {
         "content":"//username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib"
       }
       ```

     - 加工规则

       ```
       e_set("url",url_parse(v("content"),scheme="https", allow_fragments=false))
       ```

     - 加工结果

       ```
       content://username:username@example.com:8083/hello/asdah/;type=docx?
       filename=python3.docx#urllib
       url:{
              "scheme": "https",
              "netloc": "username:username@example.com:8083",
              "path": "/hello/asdah/",
       ```



```
    "params": "type=docx",
    "query": "filename=python3.docx",
    "fragment": "",
}
```

url_parse_qs

解析URL中查询字符串的组成部分。

- **函数格式**

```
url_parse_qs(
    url_qs,
    keep_blank_values=false,
    strict_parsing=false,
    encoding="utf-8",
    errors="replace",
    ignore_multi_fields=true,
)
```

- **参数说明**

参数名称	数据类型	是否必填	说明
url_qs	String	是	待解析的URL查询字符串。
keep_blank_values	Boolean	否	是否返回值为空的参数。 ● false（默认值）：不返回。 ● true：返回，且将空值处理为空字符串。
strict_parsing	Boolean	否	是否处理解析错误。 ● true：解析报错会引发ValueError异常。 ● false（默认值）：忽略错误。
encoding	String	否	指定编码方式，将含有百分号（%）的转义字符解析为Unicode字符，默认为utf-8。支持ASCII。
errors	String	否	按照编码方式无法识别字符时的处理方案。取值包括： ● ignore：直接忽略。 ● strict：直接报错，丢弃此条日志数据。 ● replace（默认值）：使用半角问号（?）替换无法识别部分。 ● xmlcharrefreplace：使用对应XML字符替换无法识别部分。

ignore_multi_fields	Num	否	指定单个返回参数的值的个数。 • true（默认值）：每个参数只返回第一个值， 类型为String。 • false：每个参数都返回所有值，类型为List。

- **返回结果**

  返回解析后的JSON数据，具体参数说明如下表所示。

字段	说明
logType	日志类型
uid	日志的唯一标识
time	日志的时间
msg	日志中的信息

- **函数示例**

  a. 示例1：设置keep_blank_values为true，返回结果中包含值为空的参数。

  - 测试数据

```
{

"content":"logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+%40+3.00GHz&deviceID=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp"
}
```

  - 加工规则

```
e_set("url",url_parse_qs(v("content"), keep_blank_values=true))
```

  - 加工结果

```
content:logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+%40+3.00GHz&deviceID=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp
 url:{
       "logType": "net_wheel_log",
       "uid": "62452****",
      "vid": "6.1.0_gf_pc",
      "asb": "1206427",
      "git": "",
```



```
    "time": "22-11-3 上11时49分33秒",
     "operatingSystem": "Windows 10  (10.0.0) 64bit",
     "deviceModel": "System Product Name (System manufacturer)",
     "graphicsDeviceName": "NVIDIA GeForce GTX 1650",
     "graphicsDeviceType": "Direct3D11",
     "graphicsDeviceVendor": "NVIDIA",
     "graphicsDeviceVersion": "Direct3D 11.0 [level 11.1]",
    "graphicsMemorySize": "3962",
     "systemMemorySize": "8127",
    "processorCount": "6",
     "processorFrequency": "3000",
     "processorType": "Intel(R) Core(TM) i5-9500F CPU @ 3.00GHz",
    "deviceID": "96da5902a042a5f84118995f88373f73650e76be166589726****",
    "guessUID": "62452****",
    "networkReachability": "wifi",
    "msg": "GetAuthkeyRsp",
  }
```


b. 示例2：设置keep_blank_values为默认值（ false ），返回结果无值为空的参数。

- 测试数据


```
{

"content":"logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+%40+3.00GHz&deviceID=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp"
}
```


- 加工规则

```
e_set("url",url_parse_qs(v("content")))
```

- 加工结果


```
content:logType=net_wheel_log&uid=62452****&vid=6.1.0_gf_pc&asb=1206427&git=&time=22-11-3+
%e4%b8%8a11%e6%97%b649%e5%88%8633%e7%a7%92&operatingSystem=Windows+10++(10.0.0)+64bit&deviceModel=System+Product+Name+(System+manufacturer)&graphicsDeviceName=NVIDIA+GeForce+GTX+1650&graphicsDeviceType=Direct3D11&graphicsDeviceVendor=NVIDIA&graphicsDeviceVersion=Direct3D+11.0+%5blevel+11.1%5d&graphicsMemorySize=3962&systemMemorySize=8127&processorCount=6&processorFrequency=3000&processorType=Intel(R)+Core(TM)+i5-9500F+CPU+%40+3.00GHz&deviceID=96da5902a042a5f84118995f88373f73650e76be166589726****&guessUID=62452****&networkReachability=wifi&msg=GetAuthkeyRsp
 url:{
      "logType": "net_wheel_log",
      "uid": "62452****",
      "vid": "6.1.0_gf_pc",
      "asb": "1206427",
     "time": "22-11-3 上11时49分33秒",
      "operatingSystem": "Windows 10  (10.0.0) 64bit",
      "deviceModel": "System Product Name (System manufacturer)",
      "graphicsDeviceName": "NVIDIA GeForce GTX 1650",
      "graphicsDeviceType": "Direct3D11",
      "graphicsDeviceVendor": "NVIDIA",
     "graphicsDeviceVersion": "Direct3D 11.0 [level 11.1]",
     "graphicsMemorySize": "3962",
      "systemMemorySize": "8127",
     "processorCount": "6",
     "processorFrequency": "3000",
      "processorType": "Intel(R) Core(TM) i5-9500F CPU @ 3.00GHz",
```



```
    "deviceID": "96da5902a042a5f84118995f88373f73650e76be166589726****",
    "guessUID": "62452****",
    "networkReachability": "wifi",
    "msg": "GetAuthkeyRsp",
  }
```

c. 示例3：设置ignore_multi_fields为默认值（true），每个参数只返回第一个值。

- 测试数据

```
{
  "content":"logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456"
}
```

- 加工规则

```
e_set("url",url_parse_qs(v("content")))
```

- 加工结果

```
content:logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456
url:{
   "logType": "net_log",
  "uid": "62452****",
   "x": ["1","2","3"],
   "asb": ["123","456"]
}
```

d. 示例4：设置ignore_multi_fields为false，每个参数返回所有值。

- 测试数据

```
{
  "content":"logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456"
}
```

- 加工规则

```
e_set("url",url_parse_qs(v("content"),ignore_multi_fields=false))
```

- 加工结果

```
content:logType=net_log&uid=62452****&x=1&x=2&x=3&asb=123&asb=456
url:{
   "logType": ["net_log"],
    "uid": ["62452****"],
    "x": ["1", "2", "3"],
    "asb": ["123", "456"],
}
```

10.2.2.2 资源函数

本文介绍资源函数的语法规则，包括参数解释、函数示例等。

函数列表

使用如下资源函数时，必须配置高级预览才能拉取到目标数据。

函数	说明
res_local	从当前数据加工任务中拉取高级参数配置信息。 支持和其他函数组合使用。
res_obs_file	从OBS中获取特定Bucket下的文件内容，支持定期更新数据。 支持和其他函数组合使用。

res_local

使用res_local函数从当前数据加工任务中拉取高级参数配置信息。支持和其他函数组合使用。

- **函数格式**
  ```
  res_local(param, default=None, type="auto")
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
param	String	是	对应高级参数配置中的Key。
default	String	否	当param参数的值不存在时，返回该参数的值，默认值为None。
type	String	否	数据输出时的数据格式。 ● auto（默认值）：将原始值转化为JSON格式。如果转换失败则返回原始值。 ● JSON：将原始值转化为JSON格式。如果转换失败则返回default参数的值。 ● raw：返回原始值。

- **返回结果**

  根据参数配置，返回JSON格式数据或者原始值。

  表 10-3 成功示例

原始值	返回值	返回值类型
1	1	整数
1.2	1.2	浮点
true	true	布尔
false	false	布尔
"123"	123	字符串
null	None	None
["v1", "v2", "v3"]	["v1", "v2", "v3"]	列表
["v1", 3, 4.0]	["v1", 3, 4.0]	列表
{"v1": 100, "v2": "good"}	{"v1": 100, "v2": "good"}	字典
{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	{"v1": {"v11": 100, "v2": 200}, "v3": "good"}	字典

表 10-4 失败示例

原始值	返回值	说明
(1,2,3)	"(1,2,3)"	不支持元组，需使用列表形式。
true	"true"	只支持true、false（小写）这两种布尔类型。
{1: 2, 3: 4}	"{1: 2, 3: 4}"	字典的关键字只能是字符串。

- **函数示例**

  从高级参数配置中获取信息并赋值给local。高级参数配置中的Key为endpoint，Value为hangzhou。

  – 测试数据

  ```
  {
    "content": "1"
  }
  ```

  – 加工规则

  ```
  e_set("local", res_local('endpoint'))
  ```

  – 加工结果

  ```
  content: 1
  local: hangzhou
  ```

res_obs_file

使用res_obs_file函数从OBS Bucket中获取文件内容，并支持定期刷新。支持和其他函数组合使用。

说明

建议日志服务Project和OBS Bucket处于同一地域，使用华为云内网获取数据。

- **函数格式**

  ```
  res_obs_file(endpoint,bucket,file)
  ```

- **参数说明**

参数名称	数据类型	是否必填	说明
endpoint	String	是	OBS中保存的文件地址。如obs.cn-north-4.huawei.com
bucket	String	是	OBS中使用的桶名，如lts-dsl。
file	String	是	目标OBS文件的路径。例如test/data.txt，不能以正斜线（/）开头。

- **返回结果**

  返回字节流形式或文本形式的文件数据。

- **函数示例**

a. 示例1：从OBS中拉取JSON格式的数据。

- JSON内容

```
{
 "users": [
  {
     "name": "user1",
     "login_historys": [
      {
        "date": "2019-10-10 0:0:0",
         "login_ip": "203.0.113.10"
      },
      {
        "date": "2019-10-10 1:0:0",
        "login_ip": "203.0.113.10"
      }
     ]
  },
  {
   "name": "user2",
     "login_historys": [
      {
        "date": "2019-10-11 0:0:0",
        "login_ip": "203.0.113.20"
      },
       {
        "date": "2019-10-11 1:0:0",
        "login_ip": "203.0.113.30"
       },
       {
        "date": "2019-10-11 1:1:0",
        "login_ip": "203.0.113.50"
       }
     ]
  }
 ]
}
```

- 测试数据

```
{
 "content":"123"
}
```

- 加工规则

```
e_set(
   "json_parse",
  json_parse(
     res_obs_file(
         "https://obs.cn-north-7.ulanqab.huawei.com",
         "lts-dsl",
         "als_dsl.json"
     )
   ),
)
```

- 加工结果

```
content: 123
   json_parse:
 '{
 "users": [
  {
     "name": "user1",
     "login_historys": [
      {
        "date": "2019-10-10 0:0:0",
        "login_ip": "203.0.113.10"
      },
       {
        "date": "2019-10-10 1:0:0",
```



```
          "login_ip": "203.0.113.10"
        }
      ]
    },
    {
      "name": "user2",
      "login_historys": [
       {
         "date": "2019-10-11 0:0:0",
         "login_ip": "203.0.113.20"
       },
       {
         "date": "2019-10-11 1:0:0",
         "login_ip": "203.0.113.30"
       },
       {
        "date": "2019-10-11 1:1:0",
        "login_ip": "203.0.113.50"
       }
      ]
    }
  ]
}'
```

b. 示例2：从OBS中拉取文本内容。

- 文本内容

```
Test bytes
```

- 测试数据

```
{
  "content": "123"
}
```

- 加工规则

```
e_set(
    "test_txt",
  res_obs_file(
          "https://obs.cn-north-7.ulanqab.huawei.com",
          "lts-dsl",
          "als_dsl.json"
      )
)
```

- 加工结果

```
content: 123
test_txt: Test bytes
```

10.2.2.3 字典函数

本文介绍字典函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
dct_make	构建字典。
dct_update	更新字典。
dct_delete	删除字典值。
dct_keys	获取字典关键字列表。

函数	说明
dct_values	获取字典值列表。
dct_get	获取字典中某关键字的值。

dct_make

构建字典。

- **函数格式**
  ```
  dct_make(key1, value1, key2, value2, ...)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
key	String	是	作为字典key的字符串。
value	String	是	作为字典value的字符串。

- **返回结果**

  返回构建的字典。
- **函数示例**
  - 测试数据
    ```
    {
      "content": "test"
    }
    ```
  - 加工规则
    ```
    e_set("hello", dct_make("k1","v1","k2","v2"))
    ```
  - 加工结果
    ```
    content:test
    hello:{"k1": "v1", "k2": "v2"}
    ```

dct_update

更新字典。

- **函数格式**
  ```
  dct_update(dict1, dict2)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
dict1	dict	是	需要被更新的目标字典。
dict2	dict	是	补充新的字典信息。

- **返回结果**

  返回更新的字典。

- **函数示例**
  - 测试数据
    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```
  - 加工规则
    ```
    e_set("hello", dct_update(v("ctx"), {"k3": "v3"}))
    ```
  - 加工结果
    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: {"k1": "v1", "k2": "v2", "k3": "v3"}
    ```

dct_delete

删除字典。

- **函数格式**
  ```
  dct_delete(dict, key1, key2, ...)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	需要删除键值对的目标字典。
key1	String	是	要删除的键值对的关键字。
key2	String	否	要删除的键值对的关键字。

- **返回结果**

  返回删除后的字典。
- **函数示例**
  - 测试数据
    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```
  - 加工规则
    ```
    e_set("hello", dct_delete(v("ctx"), "k2"))
    ```
  - 加工结果
    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: {"k1":"v1"}
    ```

dct_keys

获取字典关键字列表。

- **函数格式**
  ```
  dct_keys(dict)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。

- **返回结果**

  返回获取的字典关键词列表。

- **函数示例**

  - 测试数据

    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```

  - 加工规则

    ```
    e_set("hello", dct_keys(v("ctx")))
    ```

  - 加工结果

    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: ["k1","k2"]
    ```

dct_values

获取字典值列表。

- **函数格式**

  ```
  dct_values(dict)
  ```

- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。

- **返回结果**

  返回获取的字典词列表。

- **函数示例**

  - 测试数据

    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```

  - 加工规则

    ```
    e_set("hello", dct_values(v("ctx")))
    ```

  - 加工结果

    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: ["v1","v2"]
    ```

dct_get

获取字典中某关键字的值。

- **函数格式**

  ```
  dct_get(dict,key,default=None)
  ```

- **参数说明**

参数名称	数据类型	是否必填	说明
dict	dict	是	字典数据。
key	String	是	要获取值的关键字。

default	String	否	key不存在时，返回该值。

- **返回结果**

  返回字典关键字的值。

- **函数示例**

  a. 示例1：

  - 测试数据

    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```

  - 加工规则

    ```
    e_set("hello", dct_get(v("ctx"), "k1"))
    ```

  - 加工结果

    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: v1
    ```

  b. 示例2：

  - 测试数据

    ```
    {
      "ctx": "{\"k1\":\"v1\",\"k2\":\"v2\"}"
    }
    ```

  - 加工规则

    ```
    e_set("hello", dct_get(v("ctx"), "k3",default="123"))
    ```

  - 加工结果

    ```
    ctx: {"k1":"v1","k2":"v2"}
    hello: 123
    ```

10.2.2.4 列表函数

本文介绍列表函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
lst_make	构建列表。
lst_insert	在列表特定位置插入元素。
lst_append	在列表结尾追加元素。
lst_delete_at	在列表特定位置删除元素。
lst_reverse	反向排序列表。
lst_get	获取列表、元组中的一个元素。

lst_make

构建一个列表。

- **函数格式**
  ```
  lst_make(value1, value2, ...)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
value1	String	是	列表的元素。
value2	String	是	列表的元素。

- **返回结果**

  返回构建后的列表。
- **函数示例**
  - 测试数据
    ```
    {
      "content":"test"
    }
    ```
  - 加工规则
    ```
    e_set("hello", lst_make("k1","k2"))
    ```
  - 加工结果
    ```
    content:test
    hello:["k1", "k2"]
    ```

lst_insert

在列表特定位置插入元素。

- **函数格式**
  ```
  lst_insert(list_string, location, value1, value2, ...)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。
location	Number	是	要插入的位置。
value1	String	是	要插入的元素。
value2	String	否	要插入的元素。

- **返回结果**

  返回插入元素后的列表。
- **函数示例**
  - 测试数据
    ```
    {
      "ctx": [\"k1\",\"k2\"]
    }
    ```


- 加工规则
  ```
  e_set("hello", lst_insert(v("ctx"), 0, "k0"))
  ```
- 加工结果
  ```
  ctx: ["k1","k2"]
  hello: ["k0", "k1", "k2"]
  ```

lst_append

在列表结尾追加元素。

- **函数格式**
  ```
  lst_append(list_string, value1, value2, ...)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。
value1	String	是	要添加的元素。
value2	String	否	要添加的元素。

- **返回结果**

  返回添加元素后的列表。
- **函数示例**
  - 测试数据
    ```
    {
      "ctx": [\"k1\",\"k2\"]
    }
    ```
  - 加工规则
    ```
    e_set("hello", lst_append(v("ctx"), "k3"))
    ```
  - 加工结果
    ```
    ctx: ["k1","k2"]
    hello: ["k1", "k2", "k3"]
    ```

lst_delete_at

返回删除元素后的列表。

- **函数格式**
  ```
  lst_delete_at(list_string, location)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	list	是	传入一个列表。
location	Number	是	要删除元素的位置。第一个元素从位置0开始。

- **返回结果**

  返回获取的字典关键词列表。

- **函数示例**
  - 测试数据
    ```
    {
      "ctx": [\"k1\",\"k2\"]
    }
    ```
  - 加工规则
    ```
    e_set("hello", lst_delete_at(v("ctx"),1))
    ```
  - 加工结果
    ```
    ctx: ["k1","k2"]
    hello: ["k1"]
    ```

lst_reverse

反向排序列表。

- **函数格式**
  ```
  lst_reverse(list_string)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
list_string	List	是	传入一个列表。

- **返回结果**

  返回反转后的列表。
- **函数示例**
  - 测试数据
    ```
    {
      "ctx":[\"v1\",\"v2\"]
    }
    ```
  - 加工规则
    ```
    e_set("hello", lst_reverse(v("ctx")))
    ```
  - 加工结果
    ```
    ctx: ["v1","v2"]
    hello: ["v2","v1"]
    ```

lst_get

获取列表、元组中的一个元素。

- **函数格式**
  ```
  lst_get(list_string, location)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
ist_string	List	是	传入一个列表。
location	Int	是	从0开始计数，传入数字。例如数据为["a","b","c"]，对应获取的元素位置分别为0，1，2。

- **返回结果**

  返回列表的其中一个元素。

- **函数示例**
  - 测试数据

    ```
    {
      "ctx":[\"v1\",\"v2\"]
    }
    ```

  - 加工规则

    ```
    e_set("hello", lst_get(v("ctx"),1))
    ```

  - 加工结果

    ```
    ctx: ["v1","v2"]
    hello: "v2"
    ```

10.2.2.5 编码解码函数

本文介绍编码解码函数的语法规则，包括参数解释、函数示例等。

函数列表

分类	类型	函数	说明
编码与解码	字符串类型	**str_encode**	对数据进行编码。
		str_decode	对数据进行解码。
	Base64类型	**base64_encoding**	对数据进行Base64编码。
		base64_decoding	对数据进行Base64解码。
	HTML类型	**html_encoding**	对数据进行HTML编码。
		html_decoding	对数据进行HTML解码。
	URL类型	**url_encoding**	对数据进行URL编码。
		url_decoding	对数据进行URL解码。
压缩与解压缩	Gzip压缩库	**gzip_compress**	对数据进行压缩并编码。
		gzip_decompress	将压缩数据解压缩。
	Zlib压缩库	**zlib_compress**	对数据进行压缩并编码。
		zlib_decompress	将压缩数据解压缩。
哈希摘要	MD5哈希	**md5_encoding**	对数据进行MD5编码。

分类	类型	函数	说明
	SHA1哈希	sha1_encoding	对数据进行SHA1编码。
	循环冗余校验	crc32_encoding	计算数据的循环冗余校验码。

str_encode

按照指定的编码格式对字符串进行编码。

- **函数格式**
  ```
  str_encode(value, "utf8", errors="ignore")
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
value	任意（自动转为String）	是	需要被编码的值。
encoding	String	否	编码格式，默认为utf8。支持ASCII。
errors	String	否	按照编码格式无法识别字符时的处理方式。取值包括： • ignore（默认值）：忽略不做编码。 • strict：直接报错，丢弃此条日志数据。 • replace：使用半角问号（?）替换无法识别部分。 • xmlcharrefreplace：使用对应XML字符替换无法识别部分。

- **返回结果**

  返回编码后的字符串。
- **函数示例**

  a. 示例1：

  - 测试数据
    ```
    {
      "value": "test 测试"
    }
    ```
  - 加工规则
    ```
    e_set("result", str_encode(v("value"), "ascii", errors="ignore"))
    ```
  - 加工结果
    ```
    value: test 测试
    result: test
    ```

b. 示例2：

- 测试数据
```
{
  "value": "test 测试"
}
```
- 加工规则
```
e_set("result", str_encode(v("value"), "ascii", errors="strict"))
```
- 加工结果：执行时直接报错

c. 示例3：

- 测试数据
```
{
  "value": "test 测试"
}
```
- 加工规则
```
e_set("result", str_encode(v("value"), "ascii", errors="replace"))
```
- 加工结果
```
value: test 测试
result: test ??
```

d. 示例4：

- 测试数据
```
{
  "value": "test 测试"
}
```
- 加工规则
```
e_set("result", str_encode(v("value"), "ascii", errors="xmlcharrefreplace"))
```
- 加工结果
```
value: test 测试
result: test &#27979;&#35797;
```

str_decode

按照指定的编码格式对传入值进行解码。

- **函数格式**
```
str_decode(value, "utf8", errors="ignore")
```
- **参数说明**

参数名称	数据类型	是否必填	说明
value	任意（自动转为String）	是	需要被解码的值。
encoding	任意（自动转为String）	否	编码格式，默认为utf8。 支持ASCII。

errors	任意（自动转为String）	否	按照编码格式无法识别字符时的处理方式。取值包括： ● ignore（默认值）：忽略不做解码。 ● strict：直接报错，丢弃此条日志数据。 ● replace：使用半角问号（?）替换无法编解码部分。 ● xmlcharrefreplace：使用对应XML字符替换无法编解码部分。

- **返回结果**

  返回解码后的值。

- **函数示例**
  - 测试数据

    ```
    {
      "test": "asewds"
    }
    ```

  - 加工规则

    ```
    e_set("encoding", str_decode(b'\xe4\xbd\xa0\xe5\xa5\xbd', "utf8", 'strict'))
    ```

  - 加工结果

    ```
    test: asewds
    encoding: 你好
    ```

base64_encoding

对数据进行Base64编码。

- **函数格式**

  ```
  base64_encoding(value, format=None)
  ```

- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	填入需要被编码的值。
format	String	否	Base64编码协议。默认为format=RFC3548，还可以配置为format=RFC4648。

- **返回结果**

  返回编码后的字符串。

- **函数示例**
  - 测试数据

    ```
    {
      "str_en": "data to be encoded"
    }
    ```

  - 加工规则

    ```
    e_set("str_base64",base64_encoding(v("str_en")))
    ```


- 加工结果
  ```
  str_en : data to be encoded
  str_base64 : ZGF0YSB0byBiZSBlbmNvZGVk
  ```

base64_decoding

对数据进行Base64解码。

- **函数格式**
  ```
  base64_decoding(value, format=None)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	被解码的值。
format	String	否	Base64解码协议。默认为format=RFC3548，还可以配置为format=RFC4648。 **说明** RFC4648的Base64解码协议使用等号（=）将被解码的值填充到4字节的倍数。

- **返回结果**

  返回解码后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
      "str_de": "ZGF0YSB0byBiZSBlbmNvZGVk"
    }
    ```
  - 加工规则
    ```
    e_set("str_de_base64",base64_decoding(v("str_de")))
    ```
  - 加工结果
    ```
    str_de: ZGF0YSB0byBiZSBlbmNvZGVk
    str_de_base64: data to be encoded
    ```

html_encoding

对数据进行HTML编码。

- **函数格式**
  ```
  html_encoding(value)
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
value	String	是	被编码的值。

- **返回结果**

  返回编码后字符串。

- **函数示例**
  - 测试数据
    ```
    {
      "str":"<img class=\"size-medium wp-image-113\" style=\"margin-left: 15px;\" title=\"su1\" src=\"http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg\" alt=\"\" width=\"300\" height=\"194\" />"
    }
    ```
  - 加工规则
    ```
    e_set("str_html_en",html_encoding(v("str")))
    ```
  - 加工结果
    ```
    str : <img class="size-medium wp-image-113" style="margin-left: 15px;" title="su1" src="http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width="300" height="194" />
    str_html_en : <img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; />
    ```

html_decoding

对数据进行HTML解码。

- **函数格式**
  ```
  html_decoding(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被解码的值。

- **返回结果**

  返回解码后字符串。
- **函数示例**
  - 测试数据
    ```
    {
      "str": "<img class=&quot;size-medium wp-image-113&quot; style=&quot;margin-left: 15px;&quot; title=&quot;su1&quot; src=&quot;http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg&quot; alt=&quot;&quot; width=&quot;300&quot; height=&quot;194&quot; />"
    }
    ```
  - 加工规则
    ```
    e_set("str_html_de",html_decoding(v("str")))
    ```
  - 加工结果
    ```
    str : <img class="size-medium wp-image-113" style="margin-left: 15px;" title="su1" src="http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width="300" height="194" />
    str_html_de: <img class="size-medium wp-image-113" style="margin-left: 15px;" title="su1" src="http://aadoc.com/wp-content/uploads/2008/10/su1-300x194.jpg" alt="" width="300" height="194" />
    ```

url_encoding

对数据进行URL编码。

- **函数格式**
  ```
  url_encoding(value, plus=false)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。
plus	Boolean	否	是否将空格转换为加号，默认为false。 ● true：将空格转换为加号。 ● false：不转换空格为加号。

- **返回结果**
  返回编码后的字符串。
- **函数示例**
  a. 示例1：对数据进行URL编码。
     - 测试数据
       ```
       {
         "content": "https://www.example.org/hello/asdah"
       }
       ```
     - 加工规则
       ```
       e_set("url",url_encoding(v("content")))
       ```
     - 加工结果
       ```
       content : https://www.example.org/hello/asdah
       url: https%3A%2F%www.example.org%2FHello%2Fasdah
       ```
  b. 示例2：对数据进行URL编码。其中，未设置plus参数，不转换数据中的空格为加号。
     - 测试数据
       ```
       {
         "content": "1 2+3:4"
       }
       ```
     - 加工规则
       ```
       e_set("URL",url_encoding(v("content")))
       ```
     - 加工结果
       ```
       content:1 2+3:4
       URL:1%202%2B3%3A4
       ```
  c. 示例3：对数据进行URL编码。其中，设置plus参数为true，将数据中的空格转换为加号。
     - 测试数据
       ```
       {
         "content": "1 2+3:4"
       }
       ```
     - 加工规则
       ```
       e_set("URL", url_encoding(v("content"), plus=true))
       ```
     - 加工结果
       ```
       content:1 2+3:4
       URL:1+2%2B3%3A4
       ```


url_decoding

对数据进行URL解码。

- **函数格式**
  ```
  url_decoding(value, plus=false)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被解码的值。
plus	Boolean	否	是否将加号转换为空格，默认为false。 ● true：将加号转换为空格。 ● false：不转换加号为空格。

- **返回结果**

  返回解码后的字符串。
- **函数示例**

  a. 示例1：对数据进行URL解码。

  - 测试数据
    ```
    {
      "content": "https%3A%2F%www.example.org%2FHello%2Fasdah"
    }
    ```
  - 加工规则
    ```
    e_set("URL",url_decoding(v("content")))
    ```
  - 加工结果
    ```
    content : https%3A%2F%www.example.org%2FHello%2Fasdah
    URL : https://www.example.org/hello/asdah
    ```

  b. 示例2：对数据进行URL解码。其中，未设置plus参数，不转换数据中的加号为空格。

  - 测试数据
    ```
    {
      "content": "/answer?event_date=2022-06-30+09%3A06%3A53%20123"
    }
    ```
  - 加工规则
    ```
    e_set("URL",url_decoding(v("content")))
    ```
  - 加工结果
    ```
    content:/answer?event_date=2022-06-30+09%3A06%3A53%20123
    URL:/answer?event_date=2022-06-30+09:06:53 123
    ```

  c. 示例3：对数据进行URL解码。其中，设置plus参数为true，将数据中的加号转换为空格。

  - 测试数据
    ```
    {
      "content": "/answer?event_date=2022-06-30+09%3A06%3A53%20123"
    }
    ```
  - 加工规则


```
e_set("URL",url_decoding(v("content"),plus=true))
```

- 加工结果

```
content:/answer?event_date=2022-06-30+09%3A06%3A53%20123
URL:/answer?event_date=2022-06-30 09:06:53 123
```

gzip_compress

将数据进行压缩并编码。

- **函数格式**

```
gzip_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
data	String	是	输入需要压缩的数据。
compresslevel	Int	否	用于控制压缩等级，可配置为0~9的整数。默认值为6。 ● 1：压缩速度最快但压缩比例最小。 ● 9：压缩速度最慢但压缩比例最大。 ● 0：不压缩。
to_format	String	否	对压缩后的数据进行编码的格式，目前支持进行Base64和hex编码。
encoding	String	否	原始未压缩数据的编码格式，默认为utf-8。

- **返回结果**

  返回编码后的字符串。

- **函数示例**

  说明

  加密结果仅供参考。

  a. 示例1：对日志字段进行压缩并Base64编码。

  - 测试数据

```
{
  "content":"I always look forward to my holidays whether I travel or stay at home."
}
```

  - 加工规则

```
e_set("base64_encode_gzip_compress",gzip_compress(v("content"),to_format="base64"))
```

  - 加工结果

```
content: I always look forward to my holidays whether I travel or stay at home.
base64_encode_gzip_compress: H4sIAA8JXl4C/
xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/7wbES3rDyRsnoyQmklgNo1/
ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
```

b. 示例2：对日志字段进行hex编码。

- 测试数据
```
{
  "content":"H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNRExx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA"
}
```

- 加工规则
```
e_set("hex_encode_gzip_compress", gzip_compress(v("content"), to_format="hex"))
```

- 加工结果
```
content:H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNRExx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
hex_encode_gzip_compress:1f8b08004a478c6202ff0dc1dd0e43301800d047aa65156e3ff52f
4a2ba17649c43255194d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405
e5d1e92ef009b59c906786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a4
41bae618c679575d9bc0e306907876806c000000
```

gzip_decompress

将压缩数据解压缩。

- **函数格式**
```
gzip_decompress(data, from_format="base64", encoding="utf-8")
```
- **参数说明**

参数名称	参数类型	是否必填	说明
data	任意	是	输入需要解压的数据。
from_format	String	否	解压数据的编码格式，目前支持Base64和hex格式解码。
encoding	String	否	原始未压缩数据的编码格式，默认utf-8，其他编码格式请参见标准编码格式。

- **返回结果**

  返回解压后的对象。
- 函数示例

a. 示例1：对日志字段进行Base64解码。

- 测试数据
```
{
  "content": "H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNRExx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA"
}
```

- 加工规则
```
e_set("gzip_decompress",gzip_decompress(v("content"),from_format="base64"))
```

- 加工结果
```
content: H4sIAA8JXl4C/xXK0QmAMAwFwFXeBO7RMQKNRExx5kAZDtle/
7wbES3rDyRsnoyQmklgNo1/ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
gzip_decompress: I always look forward to my holidays whether I travel or stay at home.
```


b. 示例2：对日志字段进行hex解码。

- 测试数据

```
{
"content":"1f8b0800bff8856202ff0dc1dd0e43301800d047aa65156e3ff52f4a2ba17649c432
55194d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405e5d1e92ef009b5
9c906786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a441bae618c67957
5d9bc0e306907876806c000000"
}
```

- 加工规则

```
e_set("gzip_decompress", gzip_decompress(v("content"), from_format="hex"))
```

- 加工结果

```
content:1f8b0800bff8856202ff0dc1dd0e43301800d047aa65156e3ff52f4a2ba17649c432551
94d4a9f9e73527c64007e2e2426e81485c35628c1c42616535079bc6405e5d1e92ef009b59c9
06786a879efe1c50fb55d6c5de44cb717b2dae6d4f103f8feecbf4f88a2a441bae618c679575d9
bc0e306907876806c000000 gzip_decompress:H4sIAA8JXl4C/
xXK0QmAMAwFwFXeBO7RMQKNREx5kAZDtle/7wbES3rDyRsnoyQmklgNo1/
ztzJN08BAhjzqYGCnNCS/tPR4AcgrnWVGAAAA
```

zlib_compress

将数据进行压缩并编码。

- **函数格式**

  zlib_compress(data, compresslevel=6, to_format="base64", encoding="utf-8")

- **参数说明**

参数名称	参数类型	是否必填	说明
data	String	是	输入需要压缩的数据。
compresslevel	Int	否	用于控制压缩等级，可配置为0~9的整数。默认值为6。 ● 1：压缩速度最快但压缩比例最小。 ● 9：压缩速度最慢但压缩比例最大。 ● 0：不压缩。
to_format	String	否	对压缩后的数据进行编码的格式，目前只支持进行Base64编码。
encoding	String	否	原始未压缩数据的编码格式，默认为utf-8。

- **返回结果**

  返回编码后的对象。

- **函数示例**

  – 测试数据

  ```
  {
    "content":"I always look forward to my holidays whether I travel or stay at home."
  }
  ```

  – 加工规则


```
e_set("zlib_compress", zlib_compress(v("content"), to_format="base64"))
```

- 加工结果

  说明

  加密结果仅供参考。

```
zlib_compress: "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQlY86mBgpzQkv7T0eAGNshln"
content: "I always look forward to my holidays whether I travel or stay at home."
```

zlib_decompress

- **函数格式**

```
zlib_decompress(data, from_format="base64", encoding="utf-8")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
data	String	是	输入需要解压的数据。
from_format	String	否	解压数据的编码格式，目前只支持Base64编码。
encoding	String	否	编码格式，默认utf-8。

- **返回结果**

  返回解压后的对象。

- **函数示例**

  - 测试数据

```
{
  "content": "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQlY86mBgpzQkv7T0eAGNshln"
}
```

  - 加工规则

```
e_set("zlib_decompress", zlib_decompress(v("content"), from_format="base64"))
```

  - 加工结果

```
content: "eJwVytEJgDAMBcBV3gTu0TECjURMeZAGQ7ZXv
+8GxEt6w8kbJ6MkJpJYDaNf87cyTdPAQlY86mBgpzQkv7T0eAGNshln"
zlib_decompress: "I always look forward to my holidays whether I travel or stay at home."
```

md5_encoding

对数据进行MD5编码。

- **函数格式**

```
md5_encoding(value, format="hex")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。
format	String	否	默认值为hex，可选值：binary、hex。

- **返回结果**

  返回编码后的字符串。

- **函数示例**

  a. 示例1

  - 测试数据

    ```
    {
      "str": "GeeksforGeeks"
    }
    ```

  - 加工规则

    ```
    e_set("str_md5_en",md5_encoding(v("str")))
    ```

  - 加工结果

    ```
    str : GeeksforGeeks
    str_md5_en : f1e069787ece74531d112559945c6871
    ```

  b. 示例2

  - 测试数据

    ```
    {
      "str": "GeeksforGeeks"
    }
    ```

  - 加工规则

    ```
    e_set("str_md5_en",base64_encoding(md5_encoding(v("str"), format="binary")))
    ```

  - 加工结果

    ```
    str : GeeksforGeeks
    str_md5_en : 8eBpeH7OdFMdESVZlFxocQ==
    ```

sha1_encoding

对数据进行SHA-1编码。

- **函数格式**

  ```
  sha1_encoding(value, format=None)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	被编码的值。
format	String	否	编码类型，默认为SHA1类型，可选SHA256、SHA384、SHA224、SHA512类型。

- **返回结果**

  返回编码后的字符串。

- **函数示例**

  – 测试数据

    ```
    {
      "str": "GeeksforGeeks"
    }
    ```

  – 加工规则


```
e_set("str_sha1",sha1_encoding(v("str")))
e_set("str_sha512",sha1_encoding(v("str"),format='SHA512'))
e_set("str_sha224",sha1_encoding(v("str"),format='SHA224'))
e_set("str_sha384",sha1_encoding(v("str"),format=)))
e_set("str_sha256",sha1_encoding(v("str"),format='SHA256'))
```

- 加工结果

```
str : GeeksforGeeks
str_sha1 : 4175a37afd561152fb60c305d4fa6026b7e79856
str_sha512 :
0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75db63d1baa35ad99f2
6f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada90b722 str_sha224 :
173994f309f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d
str_sha384 :
d1e67b8819b009ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d253240490460c0a5a1a8
73da8236c12ef9b3
str_sha256 : f6071725e7ddeb434fb6b32b8ec4a2b14dd7db0d785347b2fb48f9975126178f
```

crc32_encoding

计算数据的循环冗余校验码。

- **函数格式**

```
crc32_encoding(data, input_format="raw", input_encoding="utf-8")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
data	String	是	计算校验码的值。
input_format	String	否	输入值的字符格式。默认为raw，可选值包括： ● raw：Bytes格式 ● hex：十六进制格式 ● base64：Base64编码格式
input_encoding	String	否	仅当input_format取值为raw时需要配置。配置字符编码格式。默认为utf-8。

- **返回结果**

  返回传入数据的循环冗余校验码。

- **函数示例**

  a. 示例1：计算字段test的循环冗余校验码。

  - 测试数据

```
{
  "test": "aatest"
}
```

  - 加工规则

```
e_set("str_crc32", crc32_encoding(v("test")))
```

  - 加工结果

```
str_crc32:1434103726
test:aatest
```

  b. 示例2：将字段test1和test2连接后进行MD5编码，最后计算其循环冗余校验码。

- 测试数据
```
{
"test1": "test1",
"test2": "test2"
}
```
- 加工规则
```
e_set(
    "str_crc32",
    crc32_encoding(
        md5_encoding(str_join("+", v("test1"), v("test2")), format="binary")
    ),
)
```
- 加工结果
```
str_crc32:369733261
test1:test1
test2:test2
```

c. 示例3：计算字段test的循环冗余校验码，其值为Base64编码格式。

- 测试数据
```
{
 "test": "Taloz+e+PzP3NltrEXiCig=="
}
```
- 加工规则
```
e_set("str_crc32", crc32_encoding(v("test"), input_format="base64"))
```
- 加工结果
```
str_crc32:1093789404
test:Taloz+e+PzP3NltrEXiCig==
```

10.2.2.6 IP 解析函数

本文介绍IP解析函数的语法规则，包括参数解释、函数示例等。

函数	说明
geo_parse	根据IP地址解析出所属国家、省份和市信息
ip_cidrmatch	判断IP地址是否属于CIDR地址块
ip_version	判断IP地址为IPv4还是IPv6
ip_type	判断IP地址为私有地址还是公有地址
ip_makenet	将单个IP地址转换为CIDR地址块
ip_makenet	将输入的CIDR地址块按照Prefixlen或者Netmask格式输出
ip_overlaps	判断两个网段是否存在重叠
ip2long	将字符串格式的IP地址转换成长整型数据
long2ip	将长整型数据转换成字符串格式的IP地址

geo_parse

根据IP地址解析出所属国家、省份和市信息。

- **函数格式**
  ```
  geo_parse(ip, keep_fields=None, ip_sep=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址，表示解析该IP地址所属国家、省分和市信息。如果包含多个IP地址，可通过ip_sep参数指定分割符。
keep_fields	Tuple	否	返回结果中包含的key信息。支持的key如下列表： ● city：城市名称。 ● province：省份名称。 ● country：国家名称。 ● isp：所属网络运营商名称。 ● lat：IP地址所在位置的纬度。 ● lon：IP地址所在位置的经度。 例如keep_fields=("city","country")表示仅输出city和country字段信息。 此外keep_fields也支持重命名。例如(("city","cty"),("country","state"))表示以cty和state形式输出。
ip_sep	String	否	IP地址分隔符，用于将包含多个IP地址的字符串分割为多个IP地址，解析结果通过JSON格式返回。默认值为None，表示不进行分隔。

- **返回结果**

  返回字典形式数据。
- **函数示例**

  a. 单个IP查询。

  - 测试数据
    ```
    {
      "ip": "192.168.0.1"
    }
    ```
  - 加工规则
    ```
    e_set("geo_parse", geo_parse("192.168.0.1"))
    ```
  - 加工结果
    ```
    ip: 192.168.0.1
    geo_parse: {"city": "深圳市", "province": "广东省", "country": "中国", "isp": "电信", "lat": "22.6543145968063", "lon": "114.125452397915"}
    ```

  b. 多个IP查询。

  - 测试数据


```
{
  "ip": "192.168.0.1"
}
```

- 加工规则
```
e_set("geo_parse", geo_parse("192.168.0.0,192.168.0.1,192.168.0.2", ip_sep=","))
```

- 加工结果
```
geo_parse: {"192.168.0.0": {"province": "福建省", "country": "中国", "lat": "26.07718", "lon": "119.291346"}, "192.168.0.2": {"city": "深圳市", "province": "广东省", "country": "中国", "isp": "电信", "lat": "22.6543145968063", "lon": "114.125452397915"}}
ip: 192.168.0.1
```

ip_cidrmatch

IP地址是否属于CIDR地址块。

- **函数格式**
```
ip_cidrmatch(cidr_subnet, ip, default="")
```
- **参数说明**

参数名称	参数类型	是否必填	说明
cidr_subnet	String	是	CIDR地址块
ip	String	是	IP地址。
default	String	否	如果IP地址与CIDR地址块无法匹配时，返回该值。

- **返回结果**

  当IP地址属于CIDR地址块时，返回true，否则返回false。
- **函数示例**

  a. ipv4地址与CIDR地址块匹配。

  - 测试数据
```
{
  "subnet": "192.168.1.0/24"
}
```
  - 加工规则
```
e_set("result",ip_cidrmatch(v("subnet"),"192.168.1.11"))
```
  - 加工结果
```
subnet: 192.168.1.0/24
result:    true
```

  b. ip地址与CIDR地址块无法匹配。

  - 测试数据
```
{
  "subnet": "192.168.1.0/24"
}
```
  - 加工规则
```
e_set("result",ip_cidrmatch(v("subnet"),“192.168.100.10”, default="error"))
```


- 加工结果
```
subnet: 192.168.1.0/24
result:    error
```

ip_version

判断IP地址为IPv4还是IPv6。

- **函数格式**
```
ip_version(ip, default="")
```
- **参数说明**

参数名称	参数类型	是否必填	说明
ip	String	是	输入IP地址。
default	String	否	无法判断IP地址版本时，返回该值。

- **返回结果**

  IPv4或IPv6。
- **函数示例**

  a. Ipv4地址。

  - 测试数据
```
{
  "ip": "10.21.115.10"
}
```
  - 加工规则
```
e_set("version",ip_version(v("ip")))
```
  - 加工结果
```
ip: 10.21.115.10
version: IPv4
```

  b. Ipv6地址。

  - 测试数据
```
{
  "ip": "2001:0db8:85a3:0000:0000:8a2e:0370:7334"
}
```
  - 加工规则
```
e_set("version",ip_version(v("ip")))
```
  - 加工结果
```
ip: 2001:0db8:85a3:0000:0000:8a2e:0370:7334
version: IPv6
```

ip_type

判断IP地址是私有地址还是公有地址。

- **函数格式**
```
ip_type(ip, default="")
```


- **参数说明**

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址。
default	String	否	无法判断IP地址类型时，返回该值。

- **返回结果**

  IP类型private、reserved、loopback、public和allocated ripe ncc。

- **函数示例**

  a. 私有地址。

  - 测试数据

    ```
    {
      "ip": "10.1.2.3"
    }
    ```

  - 加工规则

    ```
    e_set("result",ip_type(v("ip")))
    ```

  - 加工结果

    ```
    ip: 10.1.2.3
    result: private
    ```

  b. Ipv6地址。

  - 测试数据

    ```
    {
      "ip": "127.0.0.1"
    }
    ```

  - 加工规则

    ```
    e_set("result",ip_type(v("ip")))
    ```

  - 加工结果

    ```
    ip: 127.0.0.1
    result: loopback
    ```

ip_makenet

将单个IP地址转换为CIDR地址块。

- **函数格式**

  ```
  ip_makenet(ip, subnet_mask=None, default="")
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
ip	String	是	IP地址
subnet_mask	String	是	子网掩码， 如果ip中输入的是IP网段，则子网掩码可以为空。
default	String	否	无法将IP地址转成CIDR地址块时，返回该值。

- **返回结果**
  CIDR地址块。
- **函数示例**
  a. ip地址范围转为CIDR地址块。
     - 测试数据
       ```
       {
         "ip": "192.168.10.0-192.168.10.255"
       }
       ```
     - 加工规则
       ```
       e_set("result",ip_makenet(v("ip")))
       ```
     - 加工结果
       ```
       ip： 192.168.10.0-192.168.10.255
       result: 192.168.10.0/24
       ```
  b. ip地址转为CIDR地址块。
     - 测试数据
       ```
       {
         "ip": "192.168.10.0"
       }
       ```
     - 加工规则
       ```
       e_set("result",ip_makenet(v("ip"), “255.255.255.0”))
       ```
     - 加工结果
       ```
       ip： 192.168.10.0
       result: 192.168.10.0/24
       ```

ip_to_format

将输入的CIDR地址块按照Prefixlen或者Netmask格式输出。

- **函数格式**
  ```
  ip_to_format(cidr_subnet, want_prefix_len=0, default="")
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
cidr_subnet	String	是	输入CIDR地址块，例如：192.168.10.0/24
want_prefix_len	Int	否	设置返回格式，默认为0。 ● 0：无格式返回。 ● 1：prefix格式返回。 ● 2：netmask格式返回。 ● 3：IP网段格式返回。
default	String	否	无法将输入的CIDR地址块按照格式输出时，返回该值。

- **返回结果**
  特定格式的IP地址。

- **函数示例**

  a. 按照IP地址网段格式输出。

  - 测试数据
    ```
    {
      "ip": "192.168.11.0/24"
    }
    ```
  - 加工规则
    ```
    e_set("result",ip_to_format(v("ip"),3))
    ```
  - 加工结果
    ```
    ip: 192.168.11.0/24
    result: 192.168.11.0-192.168.11.255
    ```

  b. 按照netmask格式输出。

  - 测试数据
    ```
    {
      "ip": "192.168.11.0/24"
    }
    ```
  - 加工规则
    ```
    e_set("result",ip_to_format(v("ip"),2))
    ```
  - 加工结果
    ```
    ip: 192.168.11.0/24
    result: 192.168.11.0/255.255.255.0
    ```

  c. 按照prefix格式输出。

  - 测试数据
    ```
    {
      "ip": "192.168.11.0/24"
    }
    ```
  - 加工规则
    ```
    e_set("result",ip_to_format(v("ip"),1))
    ```
  - 加工结果
    ```
    ip: 192.168.11.0/24
    result: 192.168.11.0/24
    ```

ip_overlaps

两个网段是否存在重叠。

- **函数格式**
  ```
  ip_overlaps(cidr_subnet, cidr_subnet2, default="")
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
cidr_subnet	String	是	输入CIDR地址块1。
cidr_subnet2	String	是	输入CIDR地址块2。
default	String	否	无法判断两个CIDR地址块是否重叠时，返回该值。

- **返回结果**

  0： 两个CIDR地址块不重叠

  1： 两个CIDR地址块重叠在结束位置

  -1： 两个CIDR地址块重叠在开始位置

- **函数示例**

  a. 示例1：两个CIDR地址块不重叠。

  - 测试数据

    ```
    {
     "a": "192.168.0.0/24",
     "b": "192.168.1.0/24"
    }
    ```

  - 加工规则

    ```
    e_set("result",ip_overlaps(v("a"),v("b")))
    ```

  - 加工结果

    ```
    a: 192.168.0.0/24
    b: 192.168.1.0/24
    result: 0
    ```

  b. 示例2：两个CIDR地址在开始位置重叠。

  - 测试数据

    ```
    {
     "a": "192.168.1.0/24",
     "b": "192.168.0.0/23"
    }
    ```

  - 加工规则

    ```
    e_set("result",ip_overlaps(v("a"),v("b")))
    ```

  - 加工结果

    ```
    a: 192.168.1.0/24
    b: 192.168.0.0/23
    result: 1
    ```

  c. 示例3：两个CIDR地址在结尾位置重叠。

  - 测试数据

    ```
    {
     "a": "192.168.0.0/23",
     "b": "192.168.1.0/24"
    }
    ```

  - 加工规则

    ```
    e_set("result",ip_overlaps(v("a"),v("b")))
    ```

  - 加工结果

    ```
    a: 192.168.0.0/23
    b: 192.168.1.0/24
    result: 1
    ```

ip2long

将字符串格式的IP地址转换成长整型数据。

- **函数格式**

  ```
  ip2long(value,default=0)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的值。
default	String	否	某个不合法的IP地址被转换成的值。

- **返回结果**
  转换后的长整型数据。
- **函数示例**
  a. 正常解析。
     - 测试数据
       ```
       {
        "ip": "116.209.192.0"
       }
       ```
     - 加工规则
       ```
       e_set("result",ip2long(v("ip")))
       ```
     - 加工结果
       ```
       result: 1959903232
       ip: 116.209.192.0
       ```
  b. 非法解析。
     - 测试数据
       ```
       {
        "ip": "116.209.abc.xxx"
       }
       ```
     - 加工规则
       ```
       e_set("result",ip2long(v("ip"), "error"))
       ```
     - 加工结果
       ```
       result: error
       ip: 116.209.abc.xxx
       ```

long2ip

将长整型数据转换成字符串格式的IP地址。

- **函数格式**
  ```
  long2ip(value,default="")
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的值。
default	String	否	表示将不合法的长整型数据转成空字符串。

- **返回结果**
  长整型转换成功后的IP地址。
- **函数示例**

a. 正确的转换。

- 测试数据
```
{
 "data": "1959903232"
}
```
- 加工规则
```
e_set("ip",long2ip(v("data")))
```
- 加工结果
```
data: 1959903232
ip: 116.209.192.0
```

b. 转换失败后自定义的错误处理。

- 测试数据
```
{
 "data": "4294967296"
}
```
- 加工规则
```
e_set("ip",long2ip(v("data"), default="error"))
```
- 加工结果
```
data: 4294967296
ip: error
```

10.2.2.7 特定结构化数据函数

本文介绍特定结构化数据函数的语法规则，包括参数解释、函数示例等。

类型	函数	说明
JSON	json_select	根据JMES语法提取或计算JSON表达式中特定的值。
	json_parse	将值解析为JSON对象。
XML	xml_to_json	将xml数据转成JSON数据。

json_select

根据JMES语法提取或计算JSON表达式中特定的值。

- **函数格式**
```
json_select(value, jmes, default=None, restrict=false)
```
- **参数说明**

参数	类型	是否必填	说明
value	String、JSON	是	传入待提取字段的JSON表达式或字段。
jmes	String	是	JMES表达式，表示提取的字段。
default	String	否	如果提取字段不存在，则返回此处设置的值。默认为None，表示不返回字段。

restrict	Bool	否	提取字段的值不是合法的JSON格式时，是否严格限制加工。 默认值false。 ● false：忽略报错，数据加工继续处理，返回default定义的值。 ● true：直接报错，数据加工不再继续处理，直接丢弃该条日志。

- **返回结果**

  返回提取到的值。

- **函数示例**

  a. 示例1：从content字段提取元素name的值。

  - 测试数据

    ```
    {
      "content":  {\"name\": \"xiaoming\", \"age\": 10}
    }
    ```

  - 加工规则

    ```
    e_set("json_filter",json_select(v("content"), "name"))
    ```

  - 加工结果

    ```
    content:  {"name": "xiaoming", "age": 10}
    json_filter:  xiaoming
    ```

  b. 示例2：从content字段提取元素name包含的所有值。

  - 测试数据

    ```
    {
      "content":  {\"name\": [\"xiaoming\", \"xiaowang\", \"xiaoli\"], \"age\": 10}
    }
    ```

  - 加工规则

    ```
    e_set("json_filter", json_select(v("content"), "name[*]"))
    ```

  - 加工结果

    ```
    content:  {"name": ["xiaoming", "xiaowang", "xiaoli"], "age": 10}
    json_filter:  ["xiaoming", "xiaowang", "xiaoli"]
    ```

  c. 示例3：从content字段提取元素name3的值，若字段不存在，则返回default的值。

  - 测试数据

    ```
    {
      "content":  {\"name\": \"xiaoming\", \"age\": 10}
    }
    ```

  - 加工规则

    ```
    e_set("json_filter", json_select(v("content"), "name3", default="None"))
    ```

  - 加工结果

    ```
    content:  {"name": "xiaoming", "age": 10}
    json_filter: None
    ```

  d. 示例4：从content字段提取携带短划线（-）的元素name-test的值。

  - 测试数据

    ```
    {
      "content":  {\"name\": {\"name-test\":\"xiaoming\"}, \"age\": 10}
    }
    ```


- 加工规则
  ```
  e_set("json_filter", json_select(v("content"), 'name."name-test"', default=None))
  ```
- 加工结果
  ```
  content:  {"name": {"name-test":"xiaoming"}, "age": 10}
  json_filter: xiaoming
  ```

e. 示例5：从content字段提取携带短划线（ - ）的元素name-test的值，若元素不存在，则不返回字段。

- 测试数据
  ```
  {
    "content":  {\"name\": {\"name.test\":\"xiaoming\"}, \"age\": 10}
  }
  ```
- 加工规则
  ```
  e_set("json_filter", json_select(v("content"), 'name."name-test"', default=None))
  ```
- 加工结果
  ```
  content:  {"name": {"name-test":"xiaoming"}, "age": 10}
  ```

json_parse

将值解析为JSON对象。

- **函数格式**
  ```
  json_parse(value, default=None, restrict=false)
  ```
- **参数说明**

参数	类型	是否必填	说明
value	String	是	传入需要被解析的字段。
default	String	否	如果解析字段不存在，则返回此处设置的值。默认为None，表示不返回字段。
restrict	Bool	否	解析字段的值不是合法的JSON格式时，是否严格限制加工。 默认值false。 • false：忽略报错，数据加工继续处理，返回default定义的值。 • true：直接报错，数据加工不再继续处理，直接丢弃该条日志。

- **返回结果**

  返回转换后的JSON对象。
- **函数示例**

  a. 示例1：提取content字段的JSON值。

  - 测试数据
    ```
    {
      "content":  {\"abc\": 123, \"xyz\": \"test\" }
    }
    ```
  - 加工规则
    ```
    e_set("json", json_parse(v("content")))
    ```


- 加工结果
  content: {"abc": 123, "xyz": "test" }
  json: {"abc": 123, "xyz": "test"}

b. 示例2：提取content字段的值，如果不是JSON格式，则返回default的值。

- 测试数据
  ```
  {
    "content": "this is not json"
  }
  ```
- 加工规则
  ```
  e_set("json", json_parse(v("content"), default="FFF", restrict=false))
  ```
- 加工结果
  content: this is not json
  json: FFF

xml_to_json

将xml数据转成JSON数据。

- **函数格式**
  ```
  xml_to_json(source)
  ```
- **参数说明**

参数	类型	是否必填	说明
source	String	是	传入需要被转换的字段。

- **返回结果**

  返回转换后的JSON数据。
- **函数示例**
  - 测试数据
    ```
    {
      "str": "<data><country name=\"Liechtenstein\"><rank>1</rank><year>2008</
    year><gdppc>141100</gdppc><neighbor name=\"Austria\" direction=\"E\"/><neighbor name=
    \"Switzerland\" direction=\"W\"/></country><country name=\"Singapore\"><rank>4</
    rank><year>2011</year><gdppc>59900</gdppc><neighbor name=\"Malaysia\" direction=\"N
    \"/></country><country name=\"Panama\"><rank>68</rank><year>2011</
    year><gdppc>13600</gdppc><neighbor name=\"Costa Rica\" direction=\"W\"/><neighbor
    name=\"Colombia\" direction=\"E\"/></country></data>"
    }
    ```
  - 加工规则
    ```
    e_set("str_json",xml_to_json(v("str")))
    ```
  - 加工结果
    ```
    str:<data><country name="Liechtenstein"><rank>1</rank><year>2008</year><gdppc>141100</
    gdppc><neighbor name="Austria" direction="E"/><neighbor name="Switzerland"
    direction="W"/></country><country name="Singapore"><rank>4</rank><year>2011</
    year><gdppc>59900</gdppc><neighbor name="Malaysia" direction="N"/></country><country
    name="Panama"><rank>68</rank><year>2011</year><gdppc>13600</gdppc><neighbor
    name="Costa Rica" direction="W"/><neighbor name="Colombia" direction="E"/></country></
    data> str_json:{"data": {"country": [{"@name": "Liechtenstein", "rank": "1", "year": "2008",
    "gdppc": "141100", "neighbor": [{"@name": "Austria", "@direction": "E"}, {"@name":
    "Switzerland", "@direction": "W"}]}, {"@name": "Singapore", "rank": "4", "year": "2011",
    "gdppc": "59900", "neighbor": {"@name": "Malaysia", "@direction": "N"}}, {"@name":
    "Panama", "rank": "68", "year": "2011", "gdppc": "13600", "neighbor": [{"@name": "Costa Rica",
    "@direction": "W"}, {"@name": "Colombia", "@direction": "E"}]}]}}
    ```


10.2.2.8 正则表达式函数

本文介绍正则表达式函数的语法规则，包括参数解释、函数示例等。

类型	函数	说明
值提取函数	regex_select	根据正则表达式提取符合条件的值。
	regex_findall	根据正则表达式获得符合条件的所有值列表。
匹配判断	regex_match	判断是否匹配正则表达式。
替换	regex_replace	根据正则表达式替换字符串中的指定字符。
切分	regex_split	将一个字符串分割成字符串数组。

regex_select

- **函数格式**
  ```
  regex_select(value, r"regular expression", mi=None, gi=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	填入要匹配的值。
regular expression	String	是	匹配的正则表达式。
mi	int	否	表示匹配到的第几个表达式，默认None，表示第一个。
gi	int	否	表示匹配到的第几个分组，默认None，表示第一个。

- **返回结果**

  返回提取的值。
- **函数示例**

  a. 示例1：提取字段str中符合正则表达式的第一个值。

  - 测试数据
    ```
    {
      "str": "iZbp1a65x3r1vhpe94fi2qZ"
    }
    ```
  - 加工规则
    ```
    e_set("regex", regex_select(v("str"), r"\d+"))
    e_set("regex2", regex_select(v("str"), r"\d+", mi=None))
    e_set("regex3", regex_select(v("str"), r"\d+", mi=0))
    ```
  - 加工结果
    ```
    regex:1
    regex2:1
    ```

```
regex3:1
str:iZbp1a65x3r1vhpe94fi2qZ
```

b. 示例2：提取字段str中符合正则表达式的第一个和第二个的值。

- 测试数据

```
{
  "str": "abc123 xyz456"
}
```

- 加工规则

```
# 提取字段str中符合正则表达式的第一个值。
e_set("regex", regex_select(v("str"), r"\d+"))
# 提取字段str中符合正则表达式的第二个值。
e_set("regex2", regex_select(v("str"), r"\d+", mi=1))
```

- 加工结果

```
regex:   123
regex2:  456
str: abc123 xyz456
```

c. 示例3。

- 测试数据

```
{
  "str": "abc123 xyz456"
}
```

- 加工规则

```
# 提取字段str中符合正则表达式的第一个表达式的第一个分组的值。
e_set("regex", regex_select(v("str"),r"[a-z]+(\d+)",gi=0))
# 提取字段str中符合正则表达式的第二个表达式的第一个分组的值。
e_set("regex2", regex_select(v("str"),r"[a-z]+(\d+)",mi=1,gi=0))
# 提取字段str中符合正则表达式的第一个表达式的第一个分组的值。
e_set("regex3", regex_select(v("str"),r"([a-z]+)(\d+)",gi=0))
# 提取字段str中符合正则表达式的第一个表达式的第二个分组的值。
e_set("regex4", regex_select(v("str"),r"([a-z]+)(\d+)",gi=1))
```

- 加工结果

```
str: abc123 xyz456
regex:   123
regex2:  456
regex3:  abc
regex4:  123
```

regex_findall

根据正则表达式获得符合条件的所有值的一个列表。

- **函数格式**

```
regex_findall(value, r"regular expression")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	要匹配的值。
regular expression	String	是	正则表达式。

- **返回结果**

  返回获得符合条件的列表。

- **函数示例**

  获取字段str所有的数字。

  - 测试数据

    ```
    {
      "str": "iZbp1a65x3r1vhpe94fi2qZ"
    }
    ```

  - 加工规则

    ```
    e_set("regex_findall", regex_findall(v("str"),r"\d+"))
    ```

  - 加工结果

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    regex_findall:  ["1", "65", "3", "1", "94", "2"]
    ```

regex_match

判断是否匹配正则表达式。

- **函数格式**

  ```
  regex_match(value, r"regular expression", full=false)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	要匹配的值。
regular expression	String	是	正则表达式。
full	Bool	否	是否完全匹配，默认为false。

- **返回结果**

  返回true或者false。

- **函数示例**

  判断字段str是否包含数字。

  - 测试数据

    ```
    {
      "str": "iZbp1a65x3r1vhpe94fi2qZ"
    }
    ```

  - 加工规则

    ```
    # 判断字段str是否包含数字。
    e_set("regex_match", regex_match(v("str"),r"\d+"))
    # 判断字段str是否全部为数字。
    e_set("regex_match2", regex_match(v("str"),r"\d+",full=true))
    ```

  - 加工结果

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    regex_match:  true
    regex_match2:  false
    ```

regex_replace

根据正则表达式替换字符串中的指定字符。

- **函数格式**

  ```
  regex_replace(value, r"regular expression", replace="", count=0)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	要被替换的值。
regular expression	String	是	正则表达式。
replace	String	否	替换后的新字符。默认为空，表示删除字符。 支持正则表达式，例如r"\1****\2"，表示替换后的字符串要满足该正则表达式。 • \1代表第一个分组。 • \2代表第二个分组。
count	Number	否	替换次数。 默认为0，表示替换所有。

- **返回结果**

  返回替换后的新字符串。

- **函数示例**

  a. 示例1：把str中的所有数字用13替换。

  - 测试数据

    ```
    {
     "str": "iZbp1a65x3r1vhpe94fi2qZ ",
     "replace": "13"
    }
    ```

  - 加工规则

    ```
    e_set("regex_replace", regex_replace(v("str"),r"\d+",v("replace")))
    ```

  - 加工结果

    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    replace: 13
    regex_replace:  iZbp13a13x13r13vhpe13fi13qZ
    ```

  b. 示例2：对手机号中间4位数字进行脱敏处理。

  - 测试数据

    ```
    {
     "iphone": "13900001234"
    }
    ```

  - 加工规则

    说明

    - replace=r"\1****\2"表示替换后的字符串要满足正则表达式r"\1****\2"。
    - \1代表第一个分组，即(\d{0,3})。
    - \2代表第二个分组，即(\d{4})。

    ```
    e_set(
        "sec_iphone",
        regex_replace(v("iphone"), r"(\d{0,3})\d{4}(\d{4})", replace=r"\1****\2"),
    )
    ```

- 加工结果
  ```
  iphone: 13900001234
  sec_iphone: 139****1234
  ```

regex_split

将一个字符串分割成字符串数组。

- **函数格式**
  ```
  regex_split(value, r"regular expression", maxsplit=0)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意	是	要分裂的值。
regular expression	String	是	正则表达式。
maxsplit	int	否	最大分裂匹配次数。默认为0表示全部匹配分裂，如果为1，表示匹配中一个就分裂，剩余不再进行匹配。

- **返回结果**

  返回分割后的数组列表。
- **函数示例**

  将字段str按照数字进行分裂。
  - 测试数据
    ```
    {
      "str": "iZbp1a65x3r1vhpe94fi2qZ"
    }
    ```
  - 加工规则
    ```
    e_set("regex_split", regex_split(v("str"),r"\d+"))
    ```
  - 加工结果
    ```
    str: iZbp1a65x3r1vhpe94fi2qZ
    regex_split:  ["iZbp", "a", "x", "r", "vhpe", "fi", "qZ"]
    ```

10.2.2.9 日期时间函数

本文介绍日期时间函数的语法规则，包括参数解释、函数示例等。

在DSL加工逻辑中，日志中的所有值都以字符串的形式存储，需要根据场景对数据类型进行转换。

日志中时间主要有以下三种数据类型，您可以根据本文提供的日期时间函数进行日期时间格式转换。

- 字符串，例如2022/07/03 02-41-26。
- Unix时间戳，例如1559500886。
- 日期时间对象，例如2022-07-01 10:10:10+08:00或者2022-07-01 10:10:10。

说明

Unix时间戳本质上也是字符串。本文的日期时间函数中，除dt_parse、dt_str和dt_parsetimestamp函数支持以上三种数据类型作为参数，其他函数均需要保证参数类型的一致性。

函数列表

类型	函数	说明
通用日期时间转换	dt_parse	将值或时间表达式的值转换为日期时间对象。
	dt_str	将值或时间表达式的值转换为字符串。
	dt_parsetimestamp	将值或时间表达式的值转换为Unix时间戳。
	dt_prop	获取值或时间表达式值的特定属性，包括所属day、year等。
获取日期时间	dt_now	获取当前日期时间对象。
	dt_today	获取当前日期，不含时间。
	dt_utcnow	获取当前时区的当前日期时间对象。
	dt_fromtimestamp	将Unix时间戳转换为日期时间对象。
	dt_utcfromtimestamp	将Unix时间戳转换为当前时区的日期时间对象。
	dt_strptime	将时间字符串解析为日期时间对象。
获取Unix时间戳	dt_currentstamp	获取当前Unix时间戳。
	dt_totimestamp	将日期时间对象转换为Unix时间戳。
获取日期时间字符串	dt_strftime	将日期时间对象按照指定格式转换为字符串。
	dt_strftimestamp	将Unix时间戳按照指定格式转换为字符串。
修改日期时间	dt_truncate	从值或时间表达式中截取指定的时间粒度。
	dt_add	根据指定的时间粒度修改值或时间表达式的值。
	dt_MO	dt_add函数中传递给weekday参数的值，用于表示特定星期一的偏移量。
	dt_TU	dt_add函数中传递给weekday参数的值，用于表示特定星期二的偏移量。
	dt_WE	dt_add函数中传递给weekday参数的值，用于表示特定星期三的偏移量。

	dt_TH	dt_add函数中传递给weekday参数的值，用于表示特定星期四的偏移量。
	dt_FR	dt_add函数中传递给weekday参数的值，用于表示特定星期五的偏移量。
	dt_SA	dt_add函数中传递给weekday参数的值，用于表示特定星期六的偏移量。
	dt_SU	dt_add函数中传递给weekday参数的值，用于表示特定星期日的偏移量。
修改日期时区	dt_astimezone	将值或时间表达式的值转换为特定时区的日期时间对象。
获取差异	dt_diff	按照特定粒度获取两个值或时间表达式值的差异值。

dt_parse

将值或时间表达式的值转换为日期时间对象。

- **函数格式**
  ```
  dt_parse(value, tz=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- **返回结果**

  返回转换后的日期时间对象。
- **函数示例**

  a. 示例1：将time字段的值转化成日期时间。

  - 测试数据
    ```
    {
      "time": "1559500886"
    }
    ```
  - 加工规则
    ```
    e_set("test_time", dt_parse(v("time")))
    ```
  - 加工结果
    ```
    time: 1559500886
    test_time: 2019-06-02 18:41:26
    ```

  b. 示例2：将time字段的值转化成日期时间，时区是上海。

- 测试数据
```
{
 "time": "2019-06-01 10:10:10"
 "tz": "Asia/Shanghai"
}
```
- 加工规则
```
e_set("test_time", dt_parse(v("time"),tz=v("tz")))
```
- 加工结果
```
time: 2019-06-01 10:10:10
tz: Asia/Shanghai
test_time: 2019-06-01 10:10:10+08:00
```

dt_str

将值或时间表达式的值转换为字符串。

- **函数格式**
```
dt_str(value, fmt="format_string", tz=None)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
fmt	String	否	格式化字符串。
tz	String	否	表示时区。

- **返回结果**

  返回转换后的时间字符串。
- **函数示例**

  a. 示例1：把time字段的值转换成fmt形式，时区东京。
  - 测试数据
```
{
 "time": "2019-06-03 02:41:26",
 "fmt": "%Y/%m/%d %H-%M-%S" ,
 "tz": "Asia/Tokyo"
}
```
  - 加工规则
```
e_set("dt_str", dt_str(v("time"),fmt=v("fmt"),tz=v("tz")))
```
  - 加工结果
```
{
 "time": "2019-06-03 02:41:26"
 "fmt": "%Y/%m/%d %H-%M-%S"
 "tz": "Asia/Tokyo"
 "dt_str": "2019/06/03 02-41-26"
}
```

  b. 示例2：把time字段的值（Unix时间戳）转换成fmt形式。
  - 测试数据
```
{
 "time": "1559500886",
```



```
 "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则
```
e_set("dt_str", dt_str(v("time"),fmt=v("fmt")))
```

- 加工结果
```
time: 1559500886
fmt: %Y/%m/%d %H-%M-%S
dt_str: 2019/06/02 18-41-26
```

c. 示例3：把time字段的值转换成默认形式。

- 测试数据
```
{
 "time": "2019-06-03 02:41:26"
}
```

- 加工规则
```
e_set("dt_str", dt_str(v("time")))
```

- 加工结果
```
time: 2019-06-03 02:41:26
dt_str: 2019-06-03 02:41:26
```

dt_parsetimestamp

将值或时间表达式的值转换为Unix时间戳。

- **函数格式**
```
dt_parsetimestamp(value, tz=None)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- **返回结果**

  返回转换后的Unix时间戳。
- **函数示例**

a. 示例1：把time字段的值转换成时间戳，时区是北京。

- 测试数据
```
{
 "time": "2019-06-03 2:41:26",
 "tz": "Asia/Tokyo"
}
```

- 加工规则
```
e_set("dt_parsetimestamp", dt_parsetimestamp(v("time"),v("tz")))
```

- 加工结果


```
time: 2019-06-03 2:41:26
tz: Asia/Tokyo
dt_parsetimestamp: 1559497286
```

b. 示例2：把time字段的值转换成Unix时间戳。

- 测试数据

```
{
 "time": "2019-06-03 2:41:26"
}
```

- 加工规则

```
e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))
```

- 加工结果

```
time: 2019-06-03 2:41:26
dt_parsetimestamp: 1559529686
```

c. 示例3：把time字段的值转换成Unix时间戳。

- 测试数据

```
{
 "time": "2019-06-03 2:41:26"
}
```

- 加工规则

```
e_set("dt_parsetimestamp",dt_parsetimestamp(v("time")))
```

- 加工结果

```
time: 2019-06-03 02:41:26+8:00
dt_parsetimestamp: 1559500886
```

dt_prop

获取值或时间表达式值的特定属性，包括所属day、year等。

- **函数格式**

```
dt_prop(value, props)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
props	String	是	需要获取的时间属性。 例如属性名为year，则只会输出年份。 参数值可以为day、year、month、hour、second、minute、microsecond、weekday、weekdayname、weekdayshortname、monthname、monthshortname、dayofyear、dayofweek、weekofyear、weekofyear_m、tzname、weekofmonth。

- **返回结果**

返回提取的属性。

- **函数示例**

  a. 示例1：提取time字段的值的day属性值。

  - 测试数据
    ```
    {
     "time": "2018-10-2 09:11:40"
    }
    ```
  - 加工规则
    ```
    e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")),"day"))
    ```
  - 加工结果
    ```
    time: 2018-10-2 09:11:40
    dt_parsetimestamp: 2
    ```

  b. 示例2：提取time字段的值的year属性值。

  - 测试数据
    ```
    {
     "time": "2018-10-2 09:11:40"
    }
    ```
  - 加工规则
    ```
    e_set("dt_parsetimestamp",dt_prop(dt_parse(v("time")),"year"))
    ```
  - 加工结果
    ```
    time: 2018-10-2 09:11:40
    dt_parsetimestamp: 2018
    ```

  c. 示例3：提取time字段的值的weekdayname属性值。

  - 测试数据
    ```
    {
     "time": "2018-10-2 09:11:40"
    }
    ```
  - 加工规则
    ```
    e_set("dt_prop",dt_prop(dt_parse(v("time")),"weekdayname"))
    ```
  - 加工结果
    ```
    time: 2018-10-2 09:11:40
    dt_prop: Tuesday
    ```

  d. 示例4：提取time字段的值的weekofyear属性值。

  - 测试数据
    ```
    {
     "time": "2018-10-2 09:11:40"
    }
    ```
  - 加工规则
    ```
    e_set("dt_prop",dt_prop(dt_parse(v("time")),"weekofyear"))
    ```
  - 加工结果
    ```
    time: 2018-10-2 09:11:40
    dt_prop: 39
    ```

dt_now

获取当前日期时间对象。

- **函数格式**
  dt_now(tz=None)
- **参数说明**

参数名称	参数类型	是否必填	说明
tz	String	否	表示时区，默认为None。

- **返回结果**
  返回指定时区的时间对象。
- **函数示例**
  获取当前时间，时区是上海。
  - 测试数据
    ```
    {
      "tz": "Asia/Shanghai"
    }
    ```
  - 加工规则
    ```
    e_set("dt_now",dt_now(tz=v("tz")))
    ```
  - 加工结果
    ```
    tz: Asia/Shanghai
    dt_now: 2022-06-30 11:21:25.111836+08:00
    ```

dt_today

获取当前日期，不含时间。

- **函数格式**
  ```
  dt_today(tz=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
tz	String	否	表示时区，默认为None。

- **返回结果**
  返回指定时区的日期对象。
- **函数示例**
  获取当前日期，不含时间。
  - 测试数据
    无
  - 加工规则
    ```
    e_set("dt_today", dt_today())
    ```
  - 加工结果
    ```
    dt_today: 2022-06-30 00:00:00
    ```

dt_utcnow

获取当前UTC时间。

- **函数格式**
  ```
  dt_utcnow()
  ```
- **参数说明**

  无。
- **返回结果**

  返回当前UTC时间。
- **函数示例**

  获取当前UTC时间。
  - 测试数据

    无
  - 加工规则
    ```
    e_set("dt_utcnow",dt_utcnow())
    ```
  - 加工结果
    ```
    dt_utcnow:2022-06-30 03:33:56.614005
    ```

dt_fromtimestamp

将Unix时间戳转换为日期时间对象。

- **函数格式**
  ```
  dt_fromtimestamp(value, tz=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
tz	String	否	表示时区，默认为None。

- **返回结果**

  返回转换后的日期时间。
- **函数示例**

  a. 示例1：把time字段的值转化成日期时间对象。

  - 测试数据
    ```
    {
      "time": "1559500886"
    }
    ```
  - 加工规则
    ```
    e_set("dt_fromtimestamp",dt_fromtimestamp(v("time")))
    ```
  - 加工结果
    ```
    time: 1559500886
    dt_fromtimestamp: 2019-06-02 18:41:26
    ```

  b. 示例2：把time字段的值转化成日期时间对象，时区上海。

  - 测试数据
    ```
    {
     "time": "1559500886"
     "tz": "Asia/Shanghai"
    }
    ```

tz: Asia/Shanghai

- 加工规则
  ```
  e_set("dt_fromtimestamp",dt_fromtimestamp(v("time"),tz=v("tz")))
  ```
- 加工结果
  ```
  time: 1559500886
  tz: Asia/Shanghai
  dt_fromtimestamp: 2019-06-03 02:41:26
  ```

dt_utcfromtimestamp

将Unix时间戳转换为当前时区的日期时间对象。

- **函数格式**
  ```
  dt_utcfromtimestamp(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。

- **返回结果**

  返回转换后的日期时间对象。
- **函数示例**
  - 测试数据
    ```
    {
      "time": "1559500886"
    }
    ```
  - 加工规则
    ```
    e_set("dt_utcfromtimestamp",dt_utcfromtimestamp(v("time")))
    ```
  - 加工结果
    ```
    time: 1559500886
    dt_utcfromtimestamp: 2019-06-02 18:41:26
    ```

dt_strptime

将时间字符串解析为日期时间对象。

- **函数格式**
  ```
  dt_strptime(value, "format_string")
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
fmt	String	否	格式化字符串。

- **返回结果**

  返回解析后的日期时间对象。
- **函数示例**
  - 测试数据


```
{
 "time": "2019/06/03 02-41-26",
 "fmt": "%Y/%m/%d %H-%M-%S"
}
```

- 加工规则
  ```
  e_set("dt_strptime",dt_strptime(v("time"),v("fmt")))
  ```
- 加工结果
  ```
  time: 2019/06/03 02-41-26
  fmt: %Y/%m/%d %H-%M-%S
  dt_strptime: 2019-06-03 02:41:26
  ```

dt_currentstamp

获取当前Unix时间戳。

- **函数格式**
  ```
  dt_currentstamp(value, normalize='floor')
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	值或时间表达式。
normalize	String	否	计算结果数字格式。参数取值为： ● floor（默认值）：向下取整。 ● int：取整。 ● round：四舍五入。 ● ceil：向上取整。

- **返回结果**

  返回当前Unix时间戳。
- **函数示例**
  - 测试数据

    无
  - 加工规则
    ```
    e_set("dt_currentstamp",dt_currentstamp())
    ```
  - 加工结果
    ```
    dt_currentstamp: 1656560437
    ```

dt_totimestamp

将日期时间对象转换为Unix时间戳。

- **函数格式**
  ```
  dt_totimestamp(timeexpression)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
timeexpression	日期时间对象	是	需要被转换的日期时间对象。

- **返回结果**

  返回转换后的Unix时间戳。

- **函数示例**

  - 测试数据

    ```
    {
     "time": "2019-06-03 2:41:26"
    }
    ```

  - 加工规则

    ```
    e_set("dt_totimestamp",dt_totimestamp(dt_parse(v("time"))))
    ```

  - 加工结果

    ```
    time: 2019-06-03 2:41:26
    dt_totimestamp: 1559529686
    ```

dt_strftime

将日期时间对象按照指定格式转换为字符串。

- **函数格式**

  ```
  dt_strftime(timeexpression, "format_string")
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
timeexpression	日期时间对象	是	需要被转换的日期时间对象。
format_string	String	是	格式化字符串。

- **返回结果**

  返回格式化后的字符串。

- **函数示例**

  将time字段的值按照fmt格式转换。

  - 测试数据

    ```
    {
     "time": "2019-06-03 2:41:26",
     "fmt": "%Y/%m/%d %H-%M-%S"
    }
    ```

  - 加工规则

    ```
    e_set("dt_strftime",dt_strftime(dt_parse(v("time")),v("fmt")))
    ```

  - 加工结果

    ```
    time: 2019-06-03 2:41:26
    fmt: %Y/%m/%d %H-%M-%S
    dt_strftime: 2019/06/03 02-41-26
    ```

dt_strftimestamp

将Unix时间戳按照指定格式转换为字符串。

- **函数格式**

  ```
  dt_strftimestamp(value, fmt="format_string", tz=None)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	需要被转换的Unix时间戳。
fmt	String	是	格式化字符串。
tz	String	否	表示时区，默认为None。

- **返回结果**

  返回格式化后的字符串。

- **函数示例**

  a. 示例1

  - 测试数据

    ```
    {
     "time": "1559500886",
     "fmt": "%Y/%m/%d %H-%M-%S"
    }
    ```

  - 加工规则

    ```
    e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt")))
    ```

  - 加工结果

    ```
    time: 1559500886
    fmt: %Y/%m/%d %H-%M-%S
    dt_strftimestamp: 2019/06/02 18-41-26
    ```

  b. 示例2

  - 测试数据

    ```
    {
     "time": "1559500886",
     "fmt": "%Y/%m/%d %H-%M-%S",
     "tz": "Asia/shanghai"
    }
    ```

  - 加工规则

    ```
    e_set("dt_strftimestamp",dt_strftimestamp(v("time"),v("fmt"),v("tz")))
    ```

  - 加工结果

    ```
    dt_strftimestamp:2019/06/03 03-41-26
    fmt:%Y/%m/%d %H-%M-%S
    time:1559500886
    tz:Asia/shanghai
    ```

dt_truncate

从值或时间表达式中截取指定的时间粒度。

- **函数格式**

  ```
  dt_truncate(value, unit='day')
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明

value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
unit	String	是	需要获取的时间属性粒度。 默认为day。可选second 、minute、<num>_minute（例如：5_minute, 19_minute, 2_minute等）、hour、day、week、month、quarter、half_year、year。

- **返回结果**

  返回提取的特定时间粒度。

- **函数示例**

  a. 示例1

  - 测试数据

    ```
    {
     "time": "2019-06-03 2:41:26",
     "unit": "year"
    }
    ```

  - 加工规则

    ```
    e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
    ```

  - 加工结果

    ```
    time: 2019-06-03 2:41:26
    unit: year
    dt_truncate: 2019-01-01 00:00:00
    ```

  b. 示例2

  - 测试数据

    ```
    {
     "time": "2019-06-03 2:41:26",
     "unit": "hour"
    }
    ```

  - 加工规则

    ```
    e_set("dt_truncate",dt_truncate(v("time"),v("unit")))
    ```

  - 加工结果

    ```
    time: 2019-06-03 2:41:26
    unit: hour
    dt_truncate: 2019-06-03 02:00:00
    ```

dt_add

根据指定的时间粒度修改值或时间表达式的值。

- **函数格式**

  ```
  dt_add(value, dt1=None, dt2=None, year(s)=None, month(s)=None, day(s)=None, hour(s)=None, minute(s)=None, second(s)=None, microsecond(s)=None, week(s)=None, weekday=None)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明

value	字符串、Unix时间戳或日期时间对象	是	日期时间表达式。
dt1	字符串、Unix时间戳或日期时间对象	否	日期时间表达式，默认为None。
dt2	字符串、Unix时间戳或日期时间对象	否	日期时间表达式，默认为None。
year/years	Number	否	• year：表示需要替换的年份，例如year=2020，默认为None。 • years：表示需要增加年份的数量，如years=1表示在原来year的基础上再增加一年。
day/days	Number	否	• day：表示需要替换的天，例如day=1，默认为None。 • days：表示需要增加天的数量，如days=1表示在原来day的基础上加一天。
hour/hours	Number	否	• hour：表示需要替换的小时，例如hour=1，默认为None。 • hours：表示需要增加小时的数量，如hours=1表示在原来hour的基础上加一小时。
minute/minutes	Number	否	• minute：表示需要替换的分钟，例如minute=1，默认为None。 • minutes：表示需要增加分钟的数量，如minutes=1表示在原来minute的基础上加一分钟。
second/seconds	Number	否	• second：表示需要替换的秒数，例如second=1，默认为None。 • seconds：表示需要增加秒的数量，如seconds=1表示在原来second的基础上加一秒钟。

microsecond / microseconds	Number	否	• microsecond：表示需要替换的毫秒数，例如microsecond=1，默认为None。 • microseconds：表示需要增加毫秒的数量，microseconds=1表示在原来microsecond的基础上加一毫秒。
week/weeks	Number	否	• week：表示需要偏移的周数，例如week=1，默认为None。 • weeks：表示需要增加周的数量，weeks=1表示在原来week的基础上加一周。
weekday	Number	否	表示需要偏移的工作日，例如weekday=dt_MO(1)，默认为None。

- **返回结果**

  返回修改后的时间表达式。

- **函数示例**

  a. 示例1

  - 测试数据

    ```
    {
     "dt": "2018-10-10 1:2:3",
     "dt1": "2018-11-3 11:12:13",
     "dt2": "2018-10-1 10:10:10"
    }
    ```

  - 加工规则

    ```
    e_set("dt_add",dt_add(dt_parse(v("dt")), dt1=dt_parse(v("dt1")), dt2=dt_parse(v("dt2"))))
    ```

  - 加工结果

    ```
    dt:2018-10-10 1:2:3
    dt1:2018-11-3 11:12:13
    dt2:2018-10-1 10:10:10
    dt_add:2018-11-12 02:04:06
    ```

  b. 示例2

  - 测试数据

    ```
    {
     "dt": "2018-10-11 02:03:04",
     "year": "2019"
    }
    ```

    year: 2019

  - 加工规则

    ```
    e_set("dt_add", dt_add(dt_parse(v("dt")), year=ct_int(v("year"))))
    ```

  - 加工结果


```
dt:2018-10-11 02:03:04
dt_add:2019-10-11 02:03:04
year:2019
```

dt_MO

dt_add函数中传递给weekday参数的值，用于表示特定星期一的偏移量。

- **函数格式**
  ```
  dt_MO(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。
- **函数示例**
  - 测试数据
    ```
    {
     "time": "2019-08-13 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_MO",dt_add(v("time"),weekday=dt_MO(1)))
    ```
  - 加工结果
    ```
    time: 2019-08-13 02:03:04
    dt_MO: 2019-08-19 02:03:04
    ```

dt_TU

dt_add函数中传递给weekday参数的值，用于表示特定星期二的偏移量。

- **函数格式**
  ```
  dt_TU(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。
- **函数示例**
  - 测试数据


```
{
 "time": "2023-06-17 02:03:04"
}
```

- 加工规则
  ```
  e_set("dt_TU",dt_add(v("time"),weekday=dt_TU(1)))
  ```
- 加工结果
  ```
  time: 2023-06-17 02:03:04
  dt_TU: 2023-06-20 02:03:04
  ```

dt_WE

dt_add函数中传递给weekday参数的值，用于表示特定星期三的偏移量。

- **函数格式**
  ```
  dt_WE(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。
- **函数示例**
  - 测试数据
    ```
    {
     "time": "2023-06-17 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_WE",dt_add(v("time"),weekday=dt_WE(1)))
    ```
  - 加工结果
    ```
    time: 2023-06-17 02:03:04
    dt_WE: 2023-06-21 02:03:04
    ```

dt_TH

dt_add函数中传递给weekday参数的值，用于表示特定星期四的偏移量。

- **函数格式**
  ```
  dt_TH(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

返回偏移后的时间。

- **函数示例**
  - 测试数据
    ```
    {
     "time": "2023-06-17 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_TH",dt_add(v("time"),weekday=dt_TH(1)))
    ```
  - 加工结果
    ```
    time: 2023-06-17 02:03:04
    dt_TH: 2023-06-22 02:03:04
    ```

dt_FR

dt_add函数中传递给weekday参数的值，用于表示特定星期五的偏移量。

- **函数格式**
  ```
  dt_FR(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。
- **函数示例**
  - 测试数据
    ```
    {
     "time": "2023-06-17 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_FR",dt_add(v("time"),weekday=dt_FR(1)))
    ```
  - 加工结果
    ```
    time: 2023-06-17 02:03:04
    dt_FR: 2023-06-23 02:03:04
    ```

dt_SA

dt_add函数中传递给weekday参数的值，用于表示特定星期六的偏移量。

- **函数格式**
  ```
  dt_SA(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明

Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。

- **函数示例**
  - 测试数据
    ```
    {
     "time": "2023-06-17 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_SA",dt_add(v("time"),weekday=dt_SA(1)))
    ```
  - 加工结果
    ```
    dt_SA：2023-06-17 02:03:04
    time: 2023-06-17 02:03:04
    ```

dt_SU

dt_add函数中传递给weekday参数的值，用于表示特定星期日的偏移量。

- **函数格式**
  ```
  dt_SU(Integer_or_negative)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
Integer_or_negative	Number	是	传入偏移量。如果需要传入负数，请使用op_neg(positive) 。例如-1用op_neg(1)表示。

- **返回结果**

  返回偏移后的时间。

- **函数示例**
  - 测试数据
    ```
    {
     "time": "2023-06-17 02:03:04"
    }
    ```
  - 加工规则
    ```
    e_set("dt_SU",dt_add(v("time"),weekday=dt_SU(1)))
    ```
  - 加工结果
    ```
    dt_SU：2023-06-18 02:03:04
    time: 2023-06-17 02:03:04
    ```

dt_astimezone

将值或时间表达式的值转换为特定时区的日期时间对象。

- **函数格式**
  ```
  dt_astimezone(value, tz, reset=false)
  ```


- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
tz	String	否	表示时区，默认为None。
reset	Bool	否	表示是否对时区重新设置，默认为false表示不设置。如果为true则返回原时间加上设置的时区。

- **返回结果**

  返回特定时区的日期时间对象。

- **函数示例**

  a. 示例1

  - 测试数据

    ```
    {
     "time": "2019-06-03 2:41:26",
     "tz": "UTC"
    }
    ```

  - 加工规则

    ```
    e_set("dt_astimezone",dt_astimezone(dt_parse(v("time")), v("tz")))
    ```

  - 加工结果

    ```
    time: 2019-06-03 2:41:26
    tz: UTC
    dt_astimezone: 2019-06-03 02:41:26+00:00
    ```

  b. 示例2

  - 测试数据

    ```
    {
     "time": "2019-06-01 10:10:10+10:00",
     "tz": "Asia/shanghai"
    }
    ```

  - 加工规则

    ```
    e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=true))
    ```

  - 加工结果

    ```
    time: 2019-06-01 10:10:10+10:00
    tz: Asia/shanghai

    dt_astimezone: 2019-06-01 10:10:10+08:00
    ```

  c. 示例3

  - 测试数据

    ```
    {
     "time": "2019-06-01 10:10:10+08:00",
     "tz": "Asia/shanghai"
    }
    ```

  - 加工规则

    ```
    e_set("dt_astimezone",dt_astimezone(v("time"), v("tz"),reset=false))
    e_set("dt_astimezone_true",dt_astimezone(v("time"), v("tz"),reset=true))
    ```

- 加工结果

```
dt_astimezone: 2019-06-01 10:10:10+08:00
dt_astimezone_true: 2019-06-01 10:10:10+08:00
time:2019-06-01 10:10:10+08:00
tz:Asia/shanghai
```

dt_diff

按照特定粒度获取两个值或时间表达式值的差异值。

- **函数格式**

```
dt_diff(value1, value2, unit='second', normalize='floor')
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
value2	字符串、Unix时间戳或日期时间对象	是	值或时间表达式。
unit	String	否	按照输入时间属性返回，默认second。也可以是microsecond、millisecond、minutes、hours、day等。
normalize	String	否	计算结果数字格式。取值为： ● floor（默认值）：向下取整。 ● int：取整。 ● round：保留N位小数。 ● ceil：向上取整。

- **返回结果**

  返回按照特定粒度获取的两个值的差异值。

- **函数示例**

  a. 示例1：对time1和time2字段的值，按照秒计算差异值。

  - 测试数据

```
{
 "time1": "2018-10-1 10:10:10",
 "time2": "2018-10-1 10:10:10"
}
```

  - 加工规则

```
e_set("diff",dt_diff(v("time1"), v("time2")))
```

- 加工结果
```
time1: 2018-10-1 10:10:10
time2: 2018-10-1 10:10:10
diff: 0
```

b. 示例2：对time1和time2字段的值，按照秒计算差异值。

- 测试数据
```
{
 "time1": "2018-10-1 11:10:10",
 "time2": "2018-10-1 10:10:10"
}
```
- 加工规则
```
e_set("diff",dt_diff(v("time1"), v("time2")))
```
- 加工结果
```
time1: 2018-10-1 11:10:10
time2: 2018-10-1 10:10:10
diff: 3600
```

c. 示例3：对time1和time2字段的值，按照微秒计算差异值。

- 测试数据
```
{
 "time1": "2018-10-1 11:10:11",
 "time2": "2018-10-1 10:10:10",
 "unit": "microsecond"
}
```
- 加工规则
```
e_set("diff",dt_diff(v("time1"), v("time2"),v("unit")))
```
- 加工结果
```
diff:3601000000
time1:2018-10-1 11:10:11
time2:2018-10-1 10:10:10
unit:microsecond
```

d. 示例4：对time1和time2字段的值，按照分钟计算差异值，向下取整。

- 测试数据
```
{
 "time1": "2018-10-1 11:11:59",
  "time2": "2018-10-1 10:10:00",
  "unit": "minute ",
  "normalize": "floor"
}
```
- 加工规则
```
e_set("diff", dt_diff(v("time1"), v("time2"), v("unit"), v("normalize")))
```
- 加工结果
```
diff:61
normalize:floor
time1:2018-10-1 11:11:59
time2:2018-10-1 10:10:00
unit:minute
```

e. 示例5：对time1和time2字段的值，按照秒计算差异值，向下取整。

- 测试数据
```
{
 "time1": "10:00:00",
 "time2": "11:00:00",
 "unit": "second" ,
```



```
 "normalize": "floor"
}
```

- 加工规则

```
e_set("diff", dt_diff(v("time1"), v("time2"), v("unit"), v("normalize")))
```

加工结果

```
diff:-3600
normalize:floor
time1:10:00:00
time2:11:00:00
unit:second
```

10.2.2.10 字符串函数

10.2.2.10.1 概述

本文介绍字符串函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
多字符串操作	str_format	按照指定格式对字符串进行格式化。
	str_join	通过连接符将输入的字符串连接生成一个新的字符串。
	str_zip	将两个值或表达式的字符串进行并发分裂然后再合并成一个字符串。
排序、反转、替换	str_sort	字符串排序。
	str_reverse	将一个字符串进行反转。
	str_replace	根据规则将旧字符串替换成新字符串。
	str_translate	将字符串中的指定字符按照对应关系进行替换。
常见操作	str_strip	删除字符串中指定的字符。
	str_lstrip	删除字符串开头的指定字符。
	str_rstrip	删除字符串结尾的指定字符。
	str_lower	将字符串中所有大写字符转换为小写字符。
	str_upper	将字符串中所有小写字符转换为大写字符。
	str_title	将所有单词的第一个字母转化为大写，其余字母均为小写。
	str_capitalize	将字符串的第一个字母转化为大写，其他字母转化为小写。
	str_swapcase	对字符串的大小写字母进行转换。

类型	函数	说明
查找判断	str_count	统计字符串里某个字符出现的次数。
	str_find	判断原字符串中是否包含指定的子字符串。
	str_rfind	查找字符串中指定字符或者字符串最后一次出现的位置。
	str_endswith	判断字符串是否以指定后缀结尾。判断字符串是否以指定后缀结尾。
	str_startswith	判断字符串是否以指定字符串开头。
切分	str_split	通过指定分隔符对字符串进行分割。
	str_splitlines	通过换行符符对字符串进行分割。
	str_partition	根据指定的分隔符将字符串从左往右分割为三部分。
	str_rpartition	根据指定的分隔符将字符串从右往左分割为三部分。
格式化	str_center	用指定字符将字符串填充到指定长度。
	str_ljust	用指定字符将字符串从结尾填充至指定长度。
	str_rjust	用指定字符将原字符串从开头填充至指定长度。
	str_zfill	用字符0从开头将字符串填充至指定长度。
	str_expandtabs	将字符串中的\t转为空格。
字符集判断	str_isalnum	判断字符串是仅由字母和数字组成。
	str_isalpha	判断字符串是否仅由字母组成。
	str_isascii	判断字符串由ASCII组成。
	str_isdecimal	判断字符串是否仅包含十进制字符。
	str_isdigit	判断字符串是否仅由数字组成。
	str_isidentifier	判断字符串是否是有效的Python标识符
	str_islower	判断字符串是否由小写字母组成。
	str_isnumeric	判断字符串是否由数字组成。
	str_isprintable	判断字符串中是否所有字符都是可打印字符。
	str_isspace	判断字符串是否仅由空格字符组成。

类型	函数	说明
	str_istitle	判断字符串中所有单词的拼写首字母是否为大写，且其他字母为小写。
	str_isupper	判断字符串中所有的字母是否都为大写。
	str_uuid	随机生成UUID。

10.2.2.10.2 多字符串和排序、反转、替换

str_format

- **函数格式**
  ```
  str_format(format, str1, str2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
format	string	是	转换后的格式
str1	任意	是	待格式化的值1
str2	任意	是	待格式化的值2

- **返回结果**

  格式化后的字符串。
- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("result", str_format("{}={}", "lts", 8))
    ```
  - 加工结果
    ```
    result: lts=8
    ```

str_join

- **函数格式**
  ```
  str_join(connector, str1, str2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
connector	任意（自动转为string）	是	连接符，比如#,$,%等
str1	任意（自动转为string）	是	待连接的值1
str2	任意（自动转为string）	是	待连接的值2

- **返回结果**

  连接后的字符串。

- **函数示例**

  - 测试数据：无
  - 加工规则

    ```
    e_set("email", str_join("@", "lts", "aa", "com"))
    ```

  - 加工结果

    ```
    email:  lts@aa@com
    ```

str_zip

- **函数格式**

  ```
  str_zip(value1,value2,combine_sep=None,sep=None,quote=None,lparse=None,rparse=None)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意（自动转为String）	是	需要被合并的值。
value2	任意（自动转为String）	是	需要被合并的值。
combine_sep	任意（自动转为String）	否	合并时元素之间的合并标识，默认为**#**
sep	任意（自动转为String）	否	合并后元素之间的分隔符，仅支持单个字符，默认为**,**
quote	任意（自动转为String）	否	将合并后的元素括起来的字符，当值包含分隔符时需要使用，默认为**"**
lparse	任意（自动转为String）	否	指定value1中元素之间的分隔符和引用符，默认分隔符为,，默认引用符为"。格式为lparse=(',', '"')。 **说明** 引用符优先级高于分隔符。
rparse	任意（自动转为String）	否	指定value2中元素之间的分隔符和引用符，默认分隔符为,，默认引用符为"。格式为rparse=(',', '"')。 **说明** 引用符优先级高于分隔符。

- **返回结果**

  合并后的字符串。

- **函数示例**

  a. 示例1：sep的使用。

  - 测试数据

    ```
    {
     "key1": "value1,value11",
     "key2": "value2,value21"
    }
    ```

  - 加工规则

    ```
    e_set("combine", str_zip(v("key1"), v("key2"), sep="|"))
    ```

  - 加工结果

    ```
    key1: value1,value11
    key2: value2,value21
    combine: value1#value2|value11#value21
    ```

  b. 示例2：quote的使用。

  - 测试数据

    ```
    {
      "key1": "\"value1, value2\", value3, \"value4,value5\"",
      "key2": "value11,\"value12,value13\",value14"
    }
    ```

  - 加工规则

    ```
    e_set("combine", str_zip(v("key1"), v("key2"), quote='|'))
    ```

  - 加工结果

    ```
    key1: "value1, value2", value3, "value4,value5"
    key2: value11,"value12,value13",value14
    combine: |value1,value2#value11|,|value3#value12,value13|,|value4,value5#value14|
    ```

  c. 示例3：不同长度的值。

  - 测试数据

    ```
    {
     "key1": "value1,value2",
     "key2": "value11,value12,value13"
    }
    ```

  - 加工规则

    ```
    e_set("combine", str_zip(v("key1"), v("key2")))
    ```

  - 加工结果

    ```
    key1: value1,value2
    key2: value11,value12,value13
    combine: value1#value11,value2#value12
    ```

  d. 示例4：lparse和rparse的使用。

  - 测试数据

    ```
    {
     "key1": "|value1, value1|, value2, |value3,value3|",
     "key2": "value11, #value12,value12#, value13"
    }
    ```

  - 加工规则

    ```
    e_set("combine", str_zip(v("key1"), v("key2"), lparse=(",", '|'), rparse=(",", '#')))
    ```

  - 加工结果

    ```
    key1: |value1, value1|, value2, |value3,value3|
    key2: value11, #value12,value12#, value13
    combine: "value1,value1#value11","value2#value12,value12","value3,value3#value13"
    ```


str_sort

- **函数格式**
  ```
  str_sort(value, reverse=false)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被排序的原字符串。
reverse	Boolean	否	默认为false，表示升序排列。

- **返回结果**

  排序后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "key1": "value"
    }
    ```
  - 加工规则
    ```
    e_set("str_sort", str_sort(v("key1")))
    ```
  - 加工结果
    ```
    key1: value
    str_sort: aeluv
    ```

str_reverse

- **函数格式**
  ```
  str_reverse(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被反转的值

- **返回结果**

  反转后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
      "data": "switch"
    }
    ```
  - 加工规则
    ```
    e_set("reverse_data", str_reverse(v("data")))
    ```
  - 加工结果
    ```
    data: switch
    reverse_data: hctiws
    ```


str_replace

- **函数格式**
  ```
  str_replace(value, old, new, count)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被替换的值。
old	任意（自动转为String）	是	需要被替换的字符串。
new	任意（自动转为String）	是	替换后新的字符串。
count	Number	否	替换次数，可选项。如果不设置count，则表示替换所有。

- **返回结果**

  替换后的新字符串。
- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("str_replace", str_replace("this is string example", "is", "was"))
    ```
  - 加工结果
    ```
    str_replace： thwas was string example
    ```

str_translate

- **函数格式**
  ```
  str_translate(value, replace_string, mapping_string)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被替换的原字符串。
replace_string	任意（自动转为String）	是	需要替换的字符集合。
mapping_string	任意（自动转为String）	是	替换后的字符集合。

- **返回结果**

  处理后的字符串。
- **函数示例**
  - 测试数据：无

- 加工规则
  ```
  e_set("str_translate", str_translate("lts", "ts", "34"))
  ```
- 加工结果
  ```
  str_translate： l34
  ```

10.2.2.10.3 常见操作

str_strip

删除字符串中指定的字符。

- **函数格式**
  ```
  str_strip(value, chars)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串开头和结尾需要删除的字符集，默认为\t\r\n。

- **返回结果**

  修改后的字符串。
- **函数示例**

  a. 示例1：删除空格。

  - 测试数据
    ```
    {
      "source":"      lts"
    }
    ```
  - 加工规则
    ```
    e_set("str_strip", str_strip(v("source")))
    ```
  - 加工结果
    ```
    source:      lts
    str_strip: lts
    ```

  b. 示例2：删除开头和结尾是#的字符。

  - 测试数据
    ```
    {
      "source":  "##lts#"
    }
    ```
  - 加工规则
    ```
    e_set("str_strip", str_strip(v("source"), "#"))
    ```
  - 加工结果
    ```
    source：  ##lts#
    str_strip: lts
    ```

str_lstrip

删除字符串开头指定的字符。

- **函数格式**
  ```
  str_lstrip(value, chars)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串开头需要删除的字符集，默认为空格。

- **返回结果**

  修改后的字符串。
- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("str_strip", str_lstrip("**123**", "*"))
    ```
  - 加工结果
    ```
    str_strip: 123**
    ```

str_rstrip

删除字符串结尾指定的字符。

- **函数格式**
  ```
  str_rstrip(value, chars)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
chars	任意（自动转为String）	否	字符串结尾需要删除的字符集，默认为空格。

- **返回结果**

  修改后的字符串。
- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("str_strip", str_rstrip("**123**", "*"))
    ```
  - 加工结果
    ```
    str_strip: 123**
    ```


str_lower

将字符串中所有大写字符转换为小写字符。

- **函数格式**
  ```
  str_lower(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

  转换后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "name": "LTs"
    }
    ```
  - 加工规则
    ```
    e_set("str_lower", str_lower(v("name")))
    ```
  - 加工结果
    ```
    name: LTs
    str_lower: lts
    ```

str_upper

将字符串中所有小写字符转换为大写字符。

- **函数格式**
  ```
  str_upper(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

  转换后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "name": "LTs"
    }
    ```
  - 加工规则
    ```
    e_set("str_upper", str_upper(v("name")))
    ```
  - 加工结果
    ```
    name: LTs
    str_upper: LTS
    ```


str_title

将所有单词的第一个字母转化为大写，其余字母均为小写。

- **函数格式**
  ```
  str_title(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

  转换后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
      "name":  "for example"
    }
    ```
  - 加工规则
    ```
    e_set("str_title", str_title(v("name")))
    ```
  - 加工结果
    ```
    name:for example
    example str_title: For Exampl
    ```

str_capitalize

将字符串的第一个字母转化为大写，其他字母转化为小写。

- **函数格式**
  ```
  str_capitalize(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

  转换后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
      "value":  "welcome to xian"
    }
    ```
  - 加工规则
    ```
    e_set("str_capitalize", str_capitalize(v("value")))
    ```
  - 加工结果
    ```
    value：welcome to xian
    str_capitalize: Welcome to xian
    ```


str_swapcase

对字符串的大小写字母进行转换。

- **函数格式**
  ```
  str_swapcase(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被转换的字符串。

- **返回结果**

  转换后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "name": "this is lts"
    }
    ```
  - 加工规则
    ```
    e_set("str_swapcase", str_swapcase(v("name")))
    ```
  - 加工结果
    ```
    name: this is lts
    str_swapcase: THIS IS LTS
    ```

10.2.2.10.4 查找判断、切分和格式化

str_count

统计字符串里某个字符出现的次数。

- **函数格式**
  ```
  str_count(value, sub, start, end)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	被统计的原字符串。
sub	任意（自动转为String）	是	需要统计个数的字符。
start	Number	否	字符串开始搜索的位置。 ● 0（默认值）：第一个字符。 ● -1：最后一个字符。

end	Number	否	字符串中结束搜索的位置。 ● 0：第一个字符。 ● -1（默认值）：最后一个字符。

- **返回结果**
  返回指定字符出现的次数。
- **函数示例**
  - 测试数据
    ```
    {
      "name":  "lts is a log service"
    }
    ```
  - 加工规则
    ```
    e_set("str_count", str_count(v("name"), "l"))
    ```
  - 加工结果
    ```
    name:  lts is a log service
    str_count: 2
    ```

str_find

判断原字符串中是否包含指定的子字符串。

- **函数格式**
  ```
  str_find(value, str, begin, end)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被查找的原字符串。
str	任意（自动转为String）	是	指定查找的子字符串。
begin	Number	否	开始索引的位置。默认为0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	结束索引的位置。默认为字符串的长度。0表示第一个字符，-1表示倒数第一个字符。

- **返回结果**
  指定子字符串在原字符串中的位置。如果指定的子字符串在原字符串中出现多次，只返回第一次出现的子字符串的位置。
- **函数示例**
  - 测试数据
    ```
    {
      "name":  "lts is a log service"
    }
    ```


- 加工规则
  ```
  e_set("str_find",str_find(v("name"), "l"))
  ```
- 加工结果
  ```
  name:  lts is a log service
  str_find: 0
  ```

str_rfind

判断原字符串中是否包含指定的子字符串。

- **函数格式**
  ```
  str_rfind(value, substr, beg, end)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	被查找的原字符串。
substr	任意（自动转为String）	是	需要查找的字符。
beg	Number	否	开始查找的位置，默认为0。
end	Number	否	结束查找的位置，默认为字符串的长度。

- **返回结果**

  返回字符或字符串最后一次出现的位置。
- **函数示例**
  - 测试数据
    ```
    {
      "name":  "lts is a log service"
    }
    ```
  - 加工规则
    ```
    e_set("str_rfind", str_rfind(v("name"), "i"))
    ```
  - 加工结果
    ```
    name:  lts is a log service
    str_rfind: 17
    ```

str_endswith

判断字符串是否以指定后缀结尾。

- **函数格式**
  ```
  str_endswith(value, suffix, start, end)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被判断的原字符串。

suffix	任意（自动转为String）	是	后缀结尾规则。该参数可以是一个字符串或者是一个元素。
start	Number	否	字符串检测的起始位置。0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	字符串检测的结束位置。0表示第一个字符，-1表示倒数第一个字符。

- **返回结果**

  如果字符串以指定后缀结尾则返回true，否则返回false。

- **函数示例**
  - 测试数据
    ```
    {
      "name":  "lts is a log service"
    }
    ```
  - 加工规则
    ```
    e_set("str_endswith",str_endswith(v("name"), "service"))
    ```
  - 加工结果
    ```
    name:  lts is a log service
    str_endswith: true
    ```

str_startswith

判断字符串是否以指定字符串开头。

- **函数格式**
  ```
  str_startswith(value, prefix, start, end)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被判断的原字符串。
prefix	任意（自动转为String）	是	前缀规则。该参数可以是一个字符串或者是一个元素。
start	Number	否	字符串检测的起始位置。0表示第一个字符，-1表示倒数第一个字符。
end	Number	否	字符串检测的结束位置。0表示第一个字符，-1表示倒数第一个字符。

- **返回结果**

  如果字符串以指定前缀开头则返回true，否则返回false。

- **函数示例**

- 测试数据
```
{
  "name":  "lts is a log service"
}
```
- 加工规则
```
e_set("str_startswith",str_startswith(v("name"), "lts"))
```
- 加工结果
```
name:  lts is a log service
str_startswith: true
```

str_split

通过指定分隔符对字符串进行分割。

- **函数格式**
```
str_split(value, sep=None, maxsplit=-1)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的原字符串。
sep	Number	否	分隔符，None表示空格。
maxsplit	Number	否	最大分裂数。-1表示不限制。

- **返回结果**

  处理后的字符串。
- **函数示例**
  - 测试数据
```
{
 "content": "lts,a,log,service"
}
```
  - 加工规则
```
e_set("str_split", str_split(v("content"), ","))
```
  - 加工结果
```
content: lts,a,log,service
str_split: ["lts","a","log","service"]
```

str_splitlines

通过换行符对字符串进行分割。

- **函数格式**
```
str_splitlines(value, keepends)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的原字符串。

keepends	Boolean	否	在输出结果里是否去掉换行符（\r、\r\n、\n）。默认为false，不包含换行符，如果为true，则保留换行符。

- **返回结果**

  处理后的列表。

- **函数示例**
  - 测试数据

    ```
    {
     "value":  "lts\nis\ra\r\n"
    }
    ```

  - 加工规则

    ```
    e_set("str_splitlines", str_splitlines(v("value"), false))
    ```

  - 加工结果

    ```
    value:  lts\nis\ra\r\n
    str_splitlines: ['lts','is','a']
    ```

str_partition

根据指定的分隔符将字符串从左往右分割为三部分。

- **函数格式**

  ```
  str_partition(value, substr)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的字符串。
substr	任意（自动转为String）	否	指定的分隔符。

- **返回结果**

  分隔后的列表。

- **函数示例**
  - 测试数据

    ```
    {
     "address": "big.middle.small"
    }
    ```

  - 加工规则

    ```
    e_set("str_partition", str_partition(v("address"), "."))
    ```

  - 加工结果

    ```
    address: big.middle.small
    str_partition: ["big",".","middle.small"]
    ```

str_rpartition

根据指定的分隔符将字符串从右往左分割为三部分。

- **函数格式**
  ```
  str_rpartition(value, substr)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被分割的原字符串。
substr	任意（自动转为String）	否	指定的分隔符。

- **返回结果**

  分隔后的列表。
- **函数示例**
  - 测试数据
    ```
    {
     "address":  "big.middle.small"
    }
    ```
  - 加工规则
    ```
    e_set("str_partition", str_rpartition(v("address"), "."))
    ```
  - 加工结果
    ```
    address: big.middle.small
    str_partition: ["big.middle",".","small"]
    ```

str_center

用指定字符将字符串填充到指定长度。

- **函数格式**
  ```
  str_center(value, width, fillchar)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

  处理后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "lts is a log service"
    }
    ```
  - 加工规则
    ```
    e_set("str_center", str_center(v("value"), 40, "*"))
    ```

- 加工结果
```
center: lts is a log service
str_center: **********lts is a log service**********
```

str_ljust

用指定字符将原字符串从结尾填充至指定长度。

- **函数格式**
```
str_ljust(value, width, fillchar)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

  处理后的字符串。
- **函数示例**
  - 测试数据
```
{
 "value": "lts is a log service"
}
```
  - 加工规则
```
e_set("str_ljust", str_ljust(v("value"), 30, "*"))
```
  - 加工结果
```
value: lts is a log service
str_ljust: lts is a log service**********
```

str_rjust

用指定字符将原字符串从开头填充至指定长度。

- **函数格式**
```
str_rjust(value, width, fillchar)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。
fillchar	任意（自动转为String）	否	填充字符，默认为空格。

- **返回结果**

处理后的字符串。

- **函数示例**
  - 测试数据
    ```
    {
     "value": "lts is a log service"
    }
    ```
  - 加工规则
    ```
    e_set("str_rjust", str_rjust(v("value"), 30, "*"))
    ```
  - 加工结果
    ```
    value: lts is a log service
    str_ljust: **********lts is a log service
    ```

str_zfill

用字符0从开头将字符串填充至指定长度。

- **函数格式**
  ```
  str_zfill(value, width)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。
width	Number	是	填充后字符串的总长度。

- **返回结果**

  处理后的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "this is lts"
    }
    ```
  - 加工规则
    ```
    e_set("str_zfill", str_zfill(v("value"), 20))
    ```
  - 加工结果
    ```
    value: this is lts
    str_zfill: 000000000this is lts
    ```

str_expandtabs

将字符串中的\t转为空格。

- **函数格式**
  ```
  str_expandtabs(value, tabsize)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被修改的原字符串。

tabsize	Number	否	指定转换后空格的字符数。

- **返回结果**
  处理后的字符串。
- **函数示例**
  a. 示例1：把key中的\t字符转为空格。
     - 测试数据
       ```
       {
        "key": "this is\tlts"
       }
       ```
     - 加工规则
       ```
       e_set("str_expandtabs", str_expandtabs(v("key")))
       ```
     - 加工结果
       ```
       key: this is\tlts
       str_expandtabs: this is lts
       ```
  b. 示例2：把key中的\t字符转为空格，并填充指定空格的数量。
     - 测试数据
       ```
       {
         "key": "this is\tlts"
       }
       ```
     - 加工规则
       ```
       e_set("str_expandtabs", str_expandtabs(v("key"), 5))
       ```
     - 加工结果
       ```
       key: this is\tlts
       str_expandtabs: this is    lts
       ```

10.2.2.10.5 字符集判断

str_isalnum

判断字符串是否仅由字母和数字组成。

- **函数格式**
  ```
  str_isalnum(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**
  true/false。
- **函数示例**
  – 测试数据
    ```
    {
     "value": "ltsv5"
    }
    ```


- 加工规则
  ```
  e_set("str_isalnum", str_isalnum(v("value")))
  ```
- 加工结果
  ```
  value: ltsv5
  str_isalnum: true
  ```

str_isalpha

判断字符串是否仅由字母组成。

- **函数格式**
  ```
  str_isalpha(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "ltsv5"
    }
    ```
  - 加工规则
    ```
    e_set("str_isalpha", str_isalpha(v("value")))
    ```
  - 加工结果
    ```
    value: ltsv5
    str_isalpha: false
    ```

str_isascii

判断字符串是否在ASCII中。

- **函数格式**
  ```
  str_isascii(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "{ltsv5}#@"
    }
    ```


- 加工规则
  ```
  e_set("str_isascii", str_isascii(v("value")))
  ```
- 加工结果
  ```
  value: {ltsv5}#@
  str_isascii: true
  ```

str_isdecimal

判断字符串是否仅包含十进制字符。

- **函数格式**
  ```
  str_isdecimal(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "111"
    }
    ```
  - 加工规则
    ```
    e_set("str_isdecimal", str_isdecimal(v("value")))
    ```
  - 加工结果
    ```
    value: 111
    str_isdecimal: true
    ```

str_isdigit

判断字符串是否仅由数字组成。

- **函数格式**
  ```
  str_isdigit(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "value": "111"
    }
    ```


- 加工规则
  ```
  e_set("str_isdigit", str_isdigit(v("value")))
  ```
- 加工结果
  ```
  value: 111
  str_isdigit: true
  ```

str_isidentifier

字符串是否是有效的Python标识符，也可以用来判断变量名是否合法。

- **函数格式**
  ```
  str_isidentifier(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "key": "int"
    }
    ```
  - 加工规则
    ```
    e_set("str_isidentifier", str_isidentifier(v("key")))
    ```
  - 加工结果
    ```
    key: int
    str_isidentifier: true
    ```

str_islower

判断字符串是否由小写字母组成。

- **函数格式**
  ```
  str_islower(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "key": "lower"
    }
    ```


- 加工规则
  ```
  e_set("str_islower", str_islower(v("key")))
  ```
- 加工结果
  ```
  key: lower
  str_islower: true
  ```

str_isnumeric

判断字符串是否由数字组成。

- **函数格式**
  ```
  str_isnumeric(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "key": "123W"
    }
    ```
  - 加工规则
    ```
    e_set("str_isnumeric", str_isnumeric(v("key")))
    ```
  - 加工结果
    ```
    key: 123W
    str_isnumeric: false
    ```

str_isprintable

判断字符串中是否所有字符都是可打印字符。

- **函数格式**
  ```
  str_isprintable(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
      "key": "@#11!"
    }
    ```


- 加工规则
  ```
  e_set("str_isprintable", str_isprintable(v("key")))
  ```
- 加工结果
  ```
  key: @#11!
  str_isprintable: true
  ```

str_isspace

判断字符串是否仅由空格字符组成。

- **函数格式**
  ```
  str_isspace(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
      "key": "@#11!"
    }
    ```
  - 加工规则
    ```
    e_set("str_isspace", str_isspace(v("key")))
    ```
  - 加工结果
    ```
    key: @#11!
    str_isspace: truefalse
    ```

str_istitle

判断字符串中所有单词的拼写首字母是否为大写，且其他字母为小写。

- **函数格式**
  ```
  str_istitle(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "key":  "Lts Is A Log Service"
    }
    ```


- 加工规则
  ```
  e_set("str_istitle", str_istitle(v("key")))
  ```
- 加工结果
  ```
  key: Lts Is A Log Service
  str_istitle: true
  ```

str_isupper

判断字符串中所有的字母是否都为大写。

- **函数格式**
  ```
  str_isupper(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意（自动转为String）	是	需要被检测的字符串。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据
    ```
    {
     "key":  "LTS"
    }
    ```
  - 加工规则
    ```
    e_set("str_isupper", str_istitle(v("key")))
    ```
  - 加工结果
    ```
    key: LTS
    str_isupper: true
    ```

str_uuid

随机生成UUID。

- **函数格式**
  ```
  str_uuid(lower=true)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
lower	Boolean	否	生成的UUID中的字母是否是小写。默认为true表示小写。

- **返回结果**

  true/false。
- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("UUID", str_uuid())
    ```


- 加工结果
  UUID: 1acf7e1d-5a6b-4267-b7c1-874c8b70745d

10.2.2.11 算术函数

本文介绍算术函数的语法规则，包括参数解释、函数示例等。

函数列表

如果值为负数，请使用op_neg(positive value)函数，例如：要表示-1，请使用op_neg(1)。

表 10-5 算术函数列表

类型	函数	说明
多值计算比较	op_sum	对传入值进行求和操作。
基础计算	op_abs	对传入值进行绝对值操作。
	op_div_floor	对传入值进行整除操作。
	op_div_true	对传入值进行除法操作。
	op_pow	对值进行幂值计算操作。
	op_mul	对传入值进行乘法运算。
	op_neg	计算传入值的相反数。
	op_mod	对传入值进行模计算。
	op_sub	对传入值进行减法运算。
	op_round	对传入值进行四舍五入操作。
数学计算	mat_ceil	对传入值进行向上取整操作。
	mat_exp	以常数e为底的指数函数。
	mat_fabs	计算传入值的绝对值。
	mat_floor	对传入值进行向下取整操作。
	mat_log	计算传入值的对数。
	mat_log10	计算以10为基数的传入值的对数。

op_sum

对传入值进行求和操作。

- **函数格式**
  op_sum(value1, value2, ...)
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**
  返回多个传入值的求和结果。
- **函数示例**
  计算course_price和goods_price总价。
  - 测试数据
    ```
    {
     "course_price": 12,
     "goods_price": 2
    }
    ```
  - 加工规则
    ```
    e_set("account", op_sum(v("course_price"), v("goods_price")))
    ```
  - 加工结果
    ```
    course_price: 12
    goods_price: 2
    account: 14
    ```

op_abs

对传入值进行绝对值操作。

- **函数格式**
  ```
  op_abs(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**
  返回传入值的绝对值。
- 函数示例
  - 测试数据
    ```
    {
     "course_price": -4
    }
    ```
  - 加工规则
    ```
    e_set("op_abs", op_abs(v("course_price")))
    ```
  - 加工结果
    ```
    course_price: -4
    op_abs: 4
    ```

op_div_floor

对传入值进行整除操作。

- **函数格式**


```
op_div_floor(value1, value2)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回值1与值2整除的结果。

- **函数示例**

  根据course_price字段和count字段计算单价。

  – 测试数据

  ```
  {
   "course_price": 4,
   "count": 2
  }
  ```

  – 加工规则

  ```
  e_set("op_div_floor", op_div_floor(v("course_price"), v("count")))
  ```

  – 加工结果

  ```
  course_price: 4
  count: 2
  op_div_floor: 2
  ```

op_div_true

对传入值进行除法操作。

说明 该函数支持自动转换数据类型，输入string或者int类型的数据即可。

- **函数格式**

  ```
  op_div_true(value1, value2)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字字符串/ Number	是	待计算的值。
value2	数字字符串/ Number	是	待计算的值。

- **返回结果**

  返回值1/值2的结果。

- **函数示例**

  a. 示例1：根据fruit_price和count字段计算单价。

  ■ 测试数据

  ```
  {
   "fruit_price": 9,
   "count": 2
  }
  ```


- 加工规则
  ```
  e_set("op_div_true", op_div_true(v("fruit_price"), v("count")))
  ```
- 加工结果
  ```
  fruit_price: 9
  count: 2
  op_div_true: 4.5
  ```

b. 示例2：计算加速度（四舍五入），计算公式为a = (one_speed-two_speed)/time 。

- 测试数据
  ```
  {
   "one_speed": 9,
   "two_speed": 2,
   "time": 3
  }
  ```
- 加工规则
  ```
  e_set("a", op_round(op_div_true(op_sub(v("one_speed"), v("two_speed")), v("time")), 2))
  ```
- 加工结果
  ```
  a:2.33
  one_speed:9
  time:3
  two_speed:2
  ```

op_pow

对值进行幂值计算操作。

- **函数格式**
  ```
  op_pow(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回值1的值2次方的结果。
- **函数示例**

  计算course的pow次方的值。
  - 测试数据
    ```
    {
     "course": 100,
     "pow": 2
    }
    ```
  - 加工规则
    ```
    e_set("pow_course", op_pow(v("course"), v("pow")))
    ```
  - 加工结果
    ```
    course: 100
    pow: 2
    pow_course: 10000
    ```


op_mul

对传入值进行乘法运算。

- **函数格式**
  ```
  op_mul(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字、字符串、元祖、列表等	是	待计算的值。
value2	数字	是	待计算的值。

- **返回结果**
  - 若值1和值2为数字，返回相乘的结果。
  - 若值1和值2为字符串、元祖、列表等，返回扩大倍数的原类型。
- **函数示例**

  a. 示例1：根据course和price字段计算总价。

  - 测试数据
    ```
    {
     "course": 10,
     "price": 23
    }
    ```
  - 加工规则
    ```
    e_set("account", op_mul(ct_int(v("course")), ct_int(v("price"))))
    ```
  - 加工结果
    ```
    course: 10
    price: 23
    account: 230
    ```

  b. 示例2：将course字段扩大3倍。

  - 测试数据
    ```
    {
     "course": "abc"
    }
    ```
  - 加工规则
    ```
    e_set("course", op_mul(v("course"), 3))
    ```
  - 加工结果
    ```
    course: abcabcabc
    ```

op_neg

计算传入值的相反数。

- **函数格式**
  ```
  op_neg(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明

value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回传入值的相反数。

- **函数示例**
  - 测试数据

    ```
    {
     "course": -100
    }
    ```

  - 加工规则

    ```
    e_set("account", op_neg(v("course_price")))
    ```

  - 加工结果

    ```
    course: -100
    account: 100
    ```

op_mod

对传入值进行模计算。

- **函数格式**

  ```
  op_mod(value1, value2)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回值1对值2取模的结果。

- **函数示例**
  - 测试数据

    ```
    {
     "course": 4,
     "count": 3
    }
    ```

  - 加工规则

    ```
    e_set("op_mod", op_mod(v("course"), v("count")))
    ```

  - 加工结果

    ```
    course: 4
    count: 3
    op_mod: 1
    ```

op_sub

对传入值进行减法运算。

- **函数格式**

  ```
  op_sub(value1, value2)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回值1减去值2的结果。

- **函数示例**

  计算count减count_apple的结果。

  - 测试数据

    ```
    {
     "count": 6,
     "count_apple": 3
    }
    ```

  - 加工规则

    ```
    e_set("sub_number", op_sub(v("count"),v("count_apple")))
    ```

  - 加工结果

    ```
    count: 6
    count_apple: 3
    sub_number:  3
    ```

op_round

对传入值进行四舍五入操作。

- **函数格式**

  ```
  op_round(value, number)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。
number	数字	是	表示四舍五入后保留的小数点位数，默认值为0。

- **返回结果**

  返回传入值的四舍五入的结果。

- **函数示例**

  返回price小数点后1位数。

  - 测试数据

    ```
    {
     "price": 4.56
    }
    ```

  - 加工规则

    ```
    e_set("round_price", op_round(v("price"),1))
    ```

  - 加工结果

    ```
    price: 4.56
    round_price: 4.6
    ```


mat_ceil

对传入值进行向上取整操作。

- **函数格式**
  ```
  mat_ceil(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回大于或者等于指定值的最小整数。
- **函数示例**
  - 测试数据
    ```
    {
     "price": 4.1
    }
    ```
  - 加工规则
    ```
    e_set("mat_ceil", mat_ceil(v("price")))
    ```
  - 加工结果
    ```
    price: 4.1
    mat_ceil: 5
    ```

mat_exp

以常数e为底的指数函数，返回e的次幂结果。

- **函数格式**
  ```
  mat_exp(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回e的次幂结果。
- **函数示例**
  - 测试数据
    ```
    {
     "number": 2
    }
    ```
  - 加工规则
    ```
    e_set("e_x", mat_exp(v("number")))
    ```
  - 加工结果
    ```
    number: 2
    e_x: 7.38905609893065
    ```

mat_fabs

计算传入值的绝对值。

- **函数格式**
  ```
  mat_fabs(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回传入值的绝对值。
- **函数示例**
  - 测试数据
    ```
    {
     "course_price": -10
    }
    ```
  - 加工规则
    ```
    e_set("mat_fabs", mat_fabs(v("course_price")))
    ```
  - 加工结果
    ```
    course_price: -10
    mat_fabs: 10.0
    ```

mat_floor

对传入值进行向下取整操作。

- **函数格式**
  ```
  mat_floor(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回小于或者等于指定值的最大整数。
- **函数示例**
  - 测试数据
    ```
    {
     "course_price": 4.9
    }
    ```
  - 加工规则
    ```
    e_set("mat_floor", mat_floor(v("course_price")))
    ```
  - 加工结果
    ```
    course_price: 4.9
    mat_floor: 4
    ```

mat_log

计算传入值的对数。

- **函数格式**
  ```
  mat_log(value1,value2)
  ```


- **参数说明**

参数名称	参数类型	是否必填	说明
value1	数字或数字字符串	是	待计算的值。
value2	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回传入值的对数。

- **函数示例**
  - 测试数据

    ```
    {
     "number1": 100,
     "number2": 10
    }
    ```

  - 加工规则

    ```
    e_set("mat_log", mat_log(v("number1"),v("number2")))
    ```

  - 加工结果

    ```
    number1: 100
    number2: 10
    mat_log: 2.0
    ```

mat_log10

计算以10为基数的传入值的对数。

- **函数格式**

  ```
  mat_log10(value)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待计算的值。

- **返回结果**

  返回以10为基数的传入值的对数结果。

- **函数示例**
  - 测试数据

    ```
    {
     "number": 100
    }
    ```

  - 加工规则

    ```
    e_set("number2", mat_log10(v("number")))
    ```

  - 加工结果

    ```
    number: 100
    numbe2: 2.0
    ```

10.2.2.12 转换函数

本文主要介绍操作符函数的语法规则，包括参数说明、函数示例等。

函数列表

表 10-6 转换函数列表

类型	函数名称	功能描述
基础类型转换	ct_int	将字段或表达式的值转换为整数。
	ct_float	将字段或表达式的值转换为浮点数。
	ct_str	将字段或表达式的值转换为字符串。
	ct_bool	将字段或表达式值转换为布尔值。
数字转换	ct_chr	将字段或表达式的ANSI值、Unicode值转换为对应字符。
	ct_ord	将字段或表达式的字符转换为对应ANSI值、Unicode值。
	ct_hex	将字段或表达式的数值转换为十六进制数。
	ct_oct	将字段或表达式的数值转换为八进制数。
	ct_bin	将字段或表达式的数值转换为二进制数。
进制转换	bin2oct	将二进制数转换为八进制数。
	bin2hex	将二进制数转换为十六进制字符串。

ct_int

使用ct_int函数将字段或表达式的值转换为整数。

- **函数格式**
  ```
  ct_int(value, base=10)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。
base	Number	否	参数值所代表的进制，默认为十进制。例如base=8，表示将八进制要转成十进制。

- **返回结果**

  返回整型数值。
- **函数示例**

  a. 示例1：将字符串转换成整型。

- 测试数据
```
{
 "number": 2
}
```
- 加工规则
```
e_set("int_number", ct_int(v("number")))
```
- 加工结果
```
number: 2
int_number:  2
```

b. 示例2：将十六进制转换成十进制。

- 测试数据
```
{
 "number": AB
}
```
- 加工规则
```
e_set("int_number", ct_int(v("number"),base=16))
```
- 加工结果
```
number: AB
int_number:  171
```

ct_float

使用ct_float函数将字段或表达式的值转换为浮点数。

- **函数格式**
```
ct_float(value)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

  返回浮点类型数值。
- **函数示例**
  - 测试数据
```
{
 "price": 2
}
```
  - 加工规则
```
e_set("price_float", ct_float(v("price")))
```
  - 加工结果
```
price: 2
price_float:  2.0
```

ct_str

使用ct_str函数将字段或表达式的值转换为字符串。

- **函数格式**
```
ct_str(value)
```


- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意值	是	待转换的值。

- **返回结果**

  返回字符串。

- **函数示例**
  - 测试数据：无
  - 加工规则
    ```
    e_set("ct_str", ct_str(b'test byte'))
    ```
  - 加工结果
    ```
    ct_str: test byte
    ```

ct_bool

使用ct_bool函数将字段或表达式值转换为布尔值。

- **函数格式**
  ```
  ct_bool(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	任意值	是	待转换的值。

- **返回结果**

  返回布尔值。

- **函数示例**
  - 测试数据
    ```
    {
     "num": 2
    }
    ```
  - 加工规则
    ```
    e_set("ct_bool", ct_bool(v("num")))
    ```
  - 加工结果
    ```
    num: 2
    ct_bool: true
    ```

ct_chr

使用ct_chr函数将字段或表达式的ANSI值、Unicode值转换为对应字符。

- **函数格式**
  ```
  ct_chr(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**

  返回chr类型对应的字符。

- **函数示例**

  - 测试数据

    ```
    {
     "num": 78
    }
    ```

  - 加工规则

    ```
    e_set("ct_chr", ct_chr(v("number")))
    ```

  - 加工结果

    ```
    number: 78
    ct_chr: N
    ```

ct_ord

使用ct_ord函数将字段或表达式的字符转换为对应ANSI值、Unicode值。

- **函数格式**

  ```
  ct_ord(value)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	待转换的值，长度为1。

- **返回结果**

  返回对应的ANSI值或Unicode值。

- **函数示例**

  - 测试数据

    ```
    {
     "world": "a"
    }
    ```

  - 加工规则

    ```
    e_set("ct_ord", ct_ord(v("world")))
    ```

  - 加工结果

    ```
    world: a
    ct_ord: 97
    ```

ct_hex

使用ct_hex函数将字段或表达式的数值转换为十六进制数。

- **函数格式**

  ```
  ct_hex(value)
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**
  返回十六进制的数值。
- **函数示例**
  - 测试数据
    ```
    {
     "number": 123
    }
    ```
  - 加工规则
    ```
    e_set("ct_hex", ct_hex(v("number")))
    ```
  - 加工结果
    ```
    number: 123
    ct_hex: 0x7b
    ```

ct_oct

使用ct_oct函数将字段或表达式的数值转换为八进制数。

- **函数格式**
  ```
  ct_oct(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**
  返回八进制的数值。
- **函数示例**
  - 测试数据
    ```
    {
     "number": 123
    }
    ```
  - 加工规则
    ```
    e_set("ct_oct", ct_oct(v("number")))
    ```
  - 加工结果
    ```
    number: 123
    ct_oct: 0o173
    ```

ct_bin

使用ct_bin将字段或表达式的数值转换为二进制数。

- **函数格式**
  ```
  ct_bin(value)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	数字或数字字符串	是	待转换的值。

- **返回结果**
  返回二进制的数值。

- **函数示例**
  - 测试数据
    ```
    {
     "number": 123
    }
    ```
  - 加工规则
    ```
    e_set("ct_bin", ct_bin(v("number")))
    ```
  - 加工结果
    ```
    number: 123
    ct_bin: 0b1111011
    ```

bin2oct

使用bin2oct函数将二进制数转换为八进制数。

- **函数格式**
  ```
  bin2oct(binary)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
binary	Binary	是	Binary类型的字符串。

- **返回结果**

  返回八进制的字符串。
- **函数示例**
  - 测试数据
    ```
    {
     "test": "test"
    }
    ```
  - 加工规则
    ```
    e_set("new",bin2oct(base64_decoding("ARi8WnFiLAAACHcAGgkADV37Xs8BXftezgAdqwF9")))
    ```
  - 加工结果
    ```
    test : test
    new :
    21427426470542130000000207340006404400032567732754740127375536634000355260057 5
    ```

bin2hex

使用bin2hex函数将二进制数转换为十六进制字符串。

- **函数格式**
  ```
  bin2hex(binary)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
binary	Binary	是	Binary类型的字符串。

- **返回结果**

  返回十六进制的字符串。

- **函数示例**
  - 测试数据
    ```
    {
     "test": "test"
    }
    ```
  - 加工规则
    ```
    e_set("new",bin2hex(base64_decoding("ARi8WnFiLAAACHcAGgkADV37Xs8BXftezgAdqwF9")))
    ```
  - 加工结果
    ```
    test : test
    new :0118bc5a71622c00000877001a09000d5dfb5ecf015dfb5ece001dab017d
    ```

10.2.2.13 操作符函数

本文介绍操作符函数的语法规则，包括参数解释、函数示例等。

函数列表

如果值为负数，请使用op_neg(positive value)函数，例如：您要表示-1，请使用op_neg(1)。

表 10-7 操作符函数列表

类型	函数	说明
条件判断函数	op_if	根据判断条件返回不同表达式的值。
	op_ifnull	返回第一个值不为None的表达式的值。
	op_coalesce	返回第一个值不为None的表达式的值。
	op_nullif	如果表达式1等于表达式2，返回None。否则返回表达式1的值。
	op_and	使用逻辑运算and，对任意类型值进行真假判断，所有参数值为真时返回true。
	op_not	使用逻辑运算not，对任意类型值进行真假判断，返回参数值的反义布尔值。
	op_or	使用逻辑运算or，对任意类型值进行真假判断。当任意参数值为真时返回true，所有参数值为假时返回false。
比较	op_eq	按照a==b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_ge	按照a>=b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_gt	按照a>b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_le	按照a<=b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。

类型	函数	说明
	op_lt	按照a<b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
	op_ne	按照a!=b条件进行计算，返回true或false。a和b类型必须一致，例如都是字符串、数字或者列表。
容器判断	op_len	计算文本字符串中的字符数，可用于字符串和其他返回元组、列表、字典的表达式。
	op_in	判断字符串、元组、列表或字典中是否包含特定元素，返回true或false。
	op_not_in	判断字符串、元组、列表或字典中是否不包含特定元素，返回true或false。
	op_slice	对指定字符串、数组、元组进行截取。
	op_index	根据字符串、数组、元组的下标返回其对应的元素。
一般性多值操作	op_add	计算多个值的和，可以是字符串或者数字等。
	op_max	计算多个字段或表达式表示的数值的最大值。
	op_min	计算多个字段或表达式表示的数值的最小值。

op_if

根据判断条件返回不同表达式的值。

- **函数格式**
  ```
  op_if(condition, expression1, expression2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
condition	任意	是	判断条件。如果该条件为非布尔值，系统将对其采用真假判断。
expression1	任意	是	判断结果为true时，返回该表达式的值。
expression2	任意	是	判断结果为false时，返回该表达式的值。

- **返回结果**

  返回相应的表达式的值。
- **函数示例**

  a. 示例1：如果content为true，则把表达式1的值赋给test_if。

- 测试数据
```
{
 "content": "hello"
}
```
- 加工规则
```
e_set("test_if", op_if(v("content"),"still origion content","replace this"))
```
- 加工结果
```
content: hello
test_if:  still origion content
```

b. 示例2：如果content为false，则把表达式2的值赋给test_if。

- 测试数据
```
{
 "content": 0
}
```
- 加工规则
```
e_set("test_if", op_if(ct_int(v("content", default=0)),"still origion content","replace this"))
```
- 加工结果
```
content: 0
test_if:  replace this
```

op_ifnull

返回第一个值不为None的表达式的值。

- **函数格式**
```
op_ifnull(expression1, expression2, ....)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- **返回结果**

  返回第一个值不为None的表达式的值。
- **函数示例**

a. 示例1：

- 测试数据
```
{
 "test_if": "hello",
 "escape_name": "Etl"
}
```
- 加工规则
```
e_set("test_ifnull", op_ifnull(v("escape_name"),v("test_if")))
```
- 加工结果
```
test_if: hello
escape_name: Etl
test_ifnull:  Etl
```


b. 示例2：

- 测试数据
```
{
 "test_if": "hello",
 "escape_name": "Etl"
}
```
- 加工规则
```
e_set("test_ifnull", op_ifnull(v("test_if"),v("escape_name")))
```
- 加工结果
```
test_if: hello
escape_name: Etl
test_ifnull:  hello
```

op_coalesce

返回第一个值不为None的表达式的值。

- **函数格式**
```
op_coalesce(expression1, expression2, ...)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- **返回结果**

  返回第一个值不为None的表达式的值。
- **函数示例**

a. 示例1：

- 测试数据
```
{
 "test_if": "hello",
 "escape_name": "Etl"
}
```
- 加工规则
```
e_set("test_coalesce", op_coalesce(v("escape_name"),v("test_if")))
```
- 加工结果
```
test_if: hello
escape_name: Etl
test_coalesce: Etl
```

b. 示例2：

- 测试数据
```
{
 "test_if": "hello",
 "escape_name": "Etl"
}
```
- 加工规则
```
e_set("test_coalesce", op_coalesce(v("test_if"),v("escape_name")))
```


- 加工结果
```
test_if: hello
escape_name: Etl
test_coalesce: hello
```

op_nullif

如果表达式1等于表达式2，返回None。否则返回表达式1的值。

- **函数格式**
```
op_nullif(expression1, expression2)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- **返回结果**

  如果表达式1和表达式2相等返回None，否则返回表达式1的值。
- **函数示例**

  a. 示例1：

  - 测试数据
```
{
 "test_if": "hello",
 "escape_name": "Etl"
}
```
  - 加工规则
```
e_set("test_ifnull", op_nullif(v("test_if"),v("escape_name")))
```
  - 加工结果
```
test_if: hello
escape_name: Etl
test_ifnull:  hello
```

  b. 示例2：

  - 测试数据
```
{
 "test_if": "hello",
 "escape_name": "hello"
}
```
  - 加工规则
```
e_set("test_ifnull", op_nullif(v("content"),v("escape_name")))
```
  - 加工结果
```
#因为content与escape_name内容一样，所以没有任何内容返回给test_isnull字段。
test_if: hello
escape_name: hello
```

op_and

使用逻辑运算and，对任意类型值进行真假判断，所有参数值为真时返回true。

- **函数格式**
  ```
  op_and(value1, value2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	任意	是	运算值2。

- **返回结果**
  - 所有参数值为真时返回true。
  - 对任意类型值进行真假判断。
- **函数示例**

  a. 示例1：

  - 测试数据
    ```
    {
     "number1": 123,
     "number2": 234
    }
    ```
  - 加工规则
    ```
    e_set("op_and", op_and(v("number1"),v("number2")))
    ```
  - 加工结果
    ```
    number1: 123
    number2: 234
    op_and:  true
    ```

  b. 示例2：

  - 测试数据
    ```
    {
     "number1": 0,
     "number2": 234
    }
    ```
  - 加工规则
    ```
    e_set("op_and", op_and(v("number1"),v("number2")))
    ```
  - 加工结果
    ```
    number1: 0
    number2: 234
    op_and: false
    ```

  c. 示例3：

  - 测试数据
    ```
    {
     "ctx1": "false",
     "ctx2": 234
    }
    ```
  - 加工规则
    ```
    e_set("op_and", op_and(v("ctx1"),v("ctx2")))
    ```
  - 加工结果
    ```
    ctx1: false
    ctx2: 234
    op_and: true
    ```


d. 示例4：

- 测试数据
```
{
 "ctx1": "true",
 "ctx2": 234
}
```
- 加工规则
```
e_set("op_and", op_and(v("ctx1"),v("ctx2")))
```
- 加工结果
```
ctx1: true
ctx2: 234
op_and: true
```

op_not

使用逻辑运算not，对任意类型值进行真假判断，返回表达式值的反义布尔值。

- **函数格式**
```
op_not(expression)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
expression	任意	是	表达式。

- **返回结果**
  - 返回表达式值的反义布尔值。
  - 对任意类型值进行真假判断。
- **函数示例**

a. 示例1：

- 测试数据
```
{
 "ctx1": "true"
}
```
- 加工规则
```
e_set("op_not", op_not(v("ctx1")))
```
- 加工结果
```
ctx1: true
op_not:  false
```

b. 示例2：

- 测试数据
```
{
 "ctx1": 345
}
```
- 加工规则
```
e_set("op_not", op_not(v("ctx1")))
```
- 加工结果
```
ctx1: 345
op_not:  false
```


c. 示例3：

- 测试数据
```
{
 "ctx1": 0
}
```
- 加工规则
```
e_set("op_not", op_not(ct_int(v("ctx1"))))
```
- 加工结果
```
ctx1: 0
op_not:  true
```

d. 示例4：

- 测试数据
```
{
 "ctx1": "ETL"
}
```
- 加工规则
```
e_set("op_not", op_not(v("ctx1")))
```
- 加工结果
```
ctx1: ETL
op_not:  false
```

e. 示例5：

- 测试数据
```
{
 "ctx1": "None"
}
```
- 加工规则
```
e_set("op_not", op_not(v("ctx1")))
```
- 加工结果
```
ctx1: None
op_not:  false
```

op_or

使用逻辑运算or，对任意类型值进行真假判断。当任意表达式的值为真时返回true，所有表达式值为假时返回false。

- **函数格式**
```
op_or(expression1, expression2, ...)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
expression1	任意	是	表达式1。
expression2	任意	是	表达式2。

- **返回结果**
  - 任意表达式的值为真时返回true，所有表达式的值为假时返回false。
  - 对任意类型值进行真假判断。

- **函数示例**

  a. 示例1：

  - 测试数据

    ```
    {
     "ctx1": 123,
     "ctx2": 234
    }
    ```

  - 加工规则

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

  - 加工结果

    ```
    ctx1: 123
    ctx2: 234
    op_or:  true
    ```

  b. 示例2：

  - 测试数据

    ```
    {
     "ctx1": 0,
     "ctx2": 234
    }
    ```

  - 加工规则

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

  - 加工结果

    ```
    ctx1: 0
    ctx2: 234
    op_or:  true
    ```

  c. 示例3：

  - 测试数据

    ```
    {
     "ctx1": "ETL",
     "ctx2": "aa"
    }
    ```

  - 加工规则

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

  - 加工结果

    ```
    ctx1: ETL
    ctx2: aa
    op_or:  true
    ```

  d. 示例4：

  - 测试数据

    ```
    {
     "ctx1": "true",
     "ctx2":"false"
    }
    ```

  - 加工规则

    ```
    e_set("op_or", op_or(v("ctx1"),v("ctx2")))
    ```

  - 加工结果

    ```
    ctx1: true
    ctx2: false
    op_or:  true
    ```


e. 示例5：

- 测试数据
```
{
 "ctx1": 0,
 "ctx2":"false"
}
```
- 加工规则
```
e_set("op_or", op_or(ct_int(v("ctx1")),v("ctx2")))
```
- 加工结果
```
ctx1: 0
ctx2: false
op_or:  true
```

f. 示例6：

- 测试数据
```
{
 "ctx1": 124,
 "ctx2": "true"
}
```
- 加工规则
```
e_set("op_or", op_or(v("ctx1"),v("ctx2")))
```
- 加工结果
```
ctx1: 124
ctx2: true
op_or:  true
```

op_eq

按照a==b条件进行计算，返回true或false。

- **函数格式**
```
op_eq(value1, value2)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**

  如果值1与值2相等返回true，否则返回false。
- **函数示例**

a. 示例1：

- 测试数据
```
{
 "content": "hello",
 "ctx": "hello"
}
```
- 加工规则
```
e_set("test_eq", op_eq(v("content"),v("ctx")))
```


- 加工结果
```
content: hello
ctx: hello
test_eq: true
```

b. 示例2：

- 测试数据
```
{
 "content": "hello",
 "ctx": "ctx"
}
```
- 加工规则
```
e_set("test_eq", op_eq(v("content"),v("ctx")))
```
- 加工结果
```
content: hello
ctx: ctx
test_eq: false
```

op_ge

按照a>=b条件进行计算，返回true或false。

- **函数格式**
```
op_ge(value1, value2)
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**

  如果值1大于等于值2返回true，否则返回false。
- **函数示例**

a. 示例1：假如apple_price的值大于等于orange_price的值，则返回true。

- 测试数据
```
{
 "apple_price": 16,
 "orange_price": 14
}
```
- 加工规则
```
e_set("test_ge", op_ge(ct_int(v("apple_price")),ct_int(v("orange_price"))))
```
- 加工结果
```
apple_price: 16
orange_price: 14
test_ge: true
```

b. 示例2：假如apple_price的值小于orange_price的值，则返回false。

- 测试数据
```
{
 "apple_price": 12,
```



```
 "orange_price": 14
}
```

- 加工规则

```
e_set("test_ge", op_ge(ct_int(v("apple_price")),ct_int(v("orange_price"))))
```

- 加工结果

```
apple_price: 12
orange_price: 14
test_ge: false
```

op_gt

按照a>b条件进行计算，返回true或false。

- **函数格式**

```
op_gt(value1, value2)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**

  如果值1大于值2返回true，否则返回false。

- **函数示例**

  a. 示例1：判断old_number取值是否大于young_number取值，大于返回true否则返回false。

  - 测试数据

```
{
 "old_number": 16,
 "young_number": 14
}
```

  - 加工规则

```
e_set("op_gt",op_gt(ct_int(v("old_number")),ct_int(v("young_number"))))
```

  - 加工结果

```
old_number: 16
young_number: 14
test_ge: true
```

  b. 示例2：判断priority取值是否大于price取值，大于返回true否则返回false。

  - 测试数据

```
{
 "priority": 14,
 "price": 16
}
```

  - 加工规则

```
e_set("op_gt",op_gt(ct_int(v("priority")),ct_int(v("price"))))
```

  - 加工结果

```
priority: 14
price: 16
test_ge: false
```


op_le

按照a<=b条件进行计算，返回true或false。

- **函数格式**
  ```
  op_le(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**

  如果值1小于等于值2返回true，否则返回false。
- **函数示例**

  a. 示例1：如果priority的值小于等于price的值，返回true否则返回false。

  - 测试数据
    ```
    {
     "priority": 16,
     "price": 14
    }
    ```
  - 加工规则
    ```
    e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
    ```
  - 加工结果
    ```
    priority: 16
    price: 14
    op_le: false
    ```

  b. 示例2：如果priority的值小于等于price的值，返回true否则返回false。

  - 测试数据
    ```
    {
     "priority": 14,
     "price": 16
    }
    ```
  - 加工规则
    ```
    e_set("op_le",op_le(ct_int(v("priority")),ct_int(v("price"))))
    ```
  - 加工结果
    ```
    priority: 14
    price: 16
    test_ge: true
    ```

op_lt

按照a<b条件进行计算，返回true或false。

- **函数格式**
  ```
  op_lt(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明

value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**
  如果值1小于值2返回true，否则返回false。
- **函数示例**
  a. 示例1： 如果priority的值小于price的值，返回true否则返回false。
     - 测试数据
       ```
       {
        "priority": 16,
        "price": 14
       }
       ```
     - 加工规则
       ```
       e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
       ```
     - 加工结果
       ```
       priority: 16
       price: 14
       op_lt: false
       ```

  b. 示例2：如果priority的值小于price的值，返回true否则返回false。
     - 测试数据
       ```
       {
        "priority": 14,
        "price": 15
       }
       ```
     - 加工规则
       ```
       e_set("op_lt",op_lt(ct_int(v("priority")),ct_int(v("price"))))
       ```
     - 加工结果
       ```
       priority: 14
       price: 15
       op_lt: true
       ```

op_ne

按照a!=b条件进行计算，返回true或false。

- **函数格式**
  ```
  op_ne(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1相同	是	运算值2。

- **返回结果**
  如果值1不等于值2返回true，否则返回false。
- **函数示例**

a. 示例1：

- 测试数据
```
{
 "priority": 16,
 "price": 14
}
```
- 加工规则
```
e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
```
- 加工结果
```
priority: 16
price: 14
op_ne: true
```

b. 示例2：

- 测试数据
```
{
 "priority": 14,
 "price": 14
}
```
- 加工规则
```
e_set("op_ne",op_ne(ct_int(v("priority")),ct_int(v("price"))))
```
- 加工结果
```
priority: 14
price: 14
op_ne: false
```

op_len

计算文本字符串中的字符数，可用于字符串和其他返回元组、列表、字典的表达式。

- **函数格式**
```
op_len(value )
```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	字符串、元组、列表或字典等	是	运算值。

- **返回结果**

  返回字段的长度。
- **函数示例**
  - 测试数据
```
{
 "content": "I,love,this,world"
}
```
  - 加工规则
```
e_set("op_len",op_len(v("content")))
```
  - 加工结果
```
content: I,love,this,world
op_len: 17
```


op_in

判断字符串、元组、列表或字典中是否包含特定元素，返回true或false。

- **函数格式**
  ```
  op_in(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	字符串、元组、列表或者字典等。
value2	任意	是	判断的元素。

  说明 函数中字符串、元组、列表或字典参数在前，元素在后。
- **返回结果**
  如果字符串、元组、列表或字典a中包含元素b返回true，否则返回false。
- **函数示例**
  - 测试数据
    ```
    {
     "list":  [1, 3, 2, 7, 4, 6],
     "num2":  2
    }
    ```
  - 加工规则
    ```
    e_set("op_in",op_in(v("list"),v("num2")))
    ```
  - 加工结果
    ```
    list:  [1, 3, 2, 7, 4, 6]
    num2:  2
    op_in: true
    ```

op_not_in

判断字符串、元组、列表或字典中是否不包含特定元素，返回true或false。

- **函数格式**
  ```
  op_not_in(value1, value2)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	字符串、元组、列表或者字典等。
value2	任意	是	判断的元素。

  说明 函数中字符串、元组、列表或字典参数在前，元素在后。
- **返回结果**
  如果字符串、元组、列表或字典中不包含元素返回true，否则返回false。

- **函数示例**
  - 测试数据
    ```
    {
     "list":  [1, 3, 2, 7, 4, 6],
     "num2":  12
    }
    ```
  - 加工规则
    ```
    e_set("op_not_in",op_not_in(v("list"),v("num2")))
    ```
  - 加工结果
    ```
    list:  [1, 3, 2, 7, 4, 6]
    num2:  12
    op_not_in: true
    ```

op_slice

对指定字符串、数组、元组进行截取。

- **函数格式**
  ```
  op_slice(value, start=0, end=None, step=None)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	函数要切片的值。
start	Num	否	截取的起始位置，默认为位置0。
end	Num	否	截取的结束位置，不包含该位置，默认为字符串结尾位置。
step	Num	否	每次截取的长度。

- **返回结果**

  返回提取后的字符串。
- **函数示例**

  a. 示例1：对word字段从起点开始进行截取，结尾为2。

  - 测试数据
    ```
    {
     "word":  "I,love,this,world"
    }
    ```
  - 加工规则
    ```
    e_set("op_slice",op_slice(v("word"),2))
    ```
  - 加工结果
    ```
    word:  I,love,this,world
    op_slice: I,
    ```

  b. 示例2：对word字段从位置2到位置9进行截取，步长为1。

  - 测试数据
    ```
    {
     "word":  "I,love,this,world"
    }
    ```


- 加工规则
  ```
  e_set("op_slice",op_slice(v("word"),2,9,1))
  ```
- 加工结果
  ```
  word:  I,love,this,world
  op_slice: love,th
  ```

op_index

根据字符串、数组、元组的下标返回其对应的元素。

- **函数格式**
  ```
  op_index(value, index)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value	String	是	字符串、数组、元组等。
index	Num	否	需要传入的字符串、数组或元组的下标。

- **返回结果**

  返回下标对应的元素。
- **函数示例**

  a. 示例1：返回word字段下标为0的元素。

  - 测试数据
    ```
    {
     "word":  "I,love,this,world"
    }
    ```
  - 加工规则
    ```
    e_set("op_index",op_index(v("word"),0))
    ```
  - 加工结果
    ```
    word:  I,love,this,world
    op_index: I
    ```

  b. 示例2：返回word字段下标为3的元素。

  - 测试数据
    ```
    {
     "word":  "I,love,this,world"
    }
    ```
  - 加工规则
    ```
    e_set("op_index",op_index(v("word"),3))
    ```
  - 加工结果
    ```
    word:  I,love,this,world
    op_index: o
    ```

op_add

计算多个值的和，可以是字符串或者数字等。

- **函数格式**
  ```
  op_add(value1, value2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	字符串、元组、列表或字典等	是	运算值1。
value2	必须与值1一样	是	运算值2。

- **返回结果**

  返回求和操作后的数值。
- **函数示例**

  a. 示例1：计算price_orange和price_apple总金额。

  - 测试数据
    ```
    {
     "price_orange": 2,
     "price_apple": 13
    }
    ```
  - 加工规则
    ```
    e_set("account",op_add(ct_int(v("price_orange")),ct_int(v("price_apple"))))
    ```
  - 加工结果
    ```
     price_orange: 2,
     price_apple: 13,
     account:  15
    ```

  b. 示例2：统计bytes_in和bytes_out的和。

  - 测试数据
    ```
    {
     "bytes_in": 214,
     "bytes_out": 123
    }
    ```
  - 加工规则
    ```
    e_set("total_bytes", op_add(ct_int(v("bytes_in")), ct_int(v("bytes_out"))))
    ```
  - 加工结果
    ```
    bytes_in: 214
    bytes_out: 123
    total_bytes:  337
    ```

  c. 示例3：给网址添加HTTPS头。

  - 测试数据
    ```
    {
     "host": "xx.com"
    }
    ```
  - 加工规则
    ```
    e_set("website", op_add("https://", v("host")))
    ```
  - 加工结果
    ```
    host: xx.com
    website: https://xx.com
    ```


op_max

计算多个字段或表达式表示的数值的最大值。

- **函数格式**
  ```
  op_max(value1, value2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1一样	是	运算值2。

- **返回结果**

  返回多个数值中的最大值。
- **函数示例**
  - 测试数据
    ```
    {
     "price_orange":  2,
     "priority_apple":  13
    }
    ```
  - 加工规则
    ```
    e_set("max_price", op_max(ct_int(v("price_orange")),ct_int(v("priority_apple"))))
    ```
  - 加工结果
    ```
    price_orange:  2
    priority_apple:  13
    max_price:  13
    ```

op_min

计算多个字段或表达式表示的数值的最小值。

- **函数格式**
  ```
  op_min(value1, value2, ...)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
value1	任意	是	运算值1。
value2	必须与值1一样	是	运算值2。

- **返回结果**

  返回多个数值中的最小值。
- **函数示例**
  - 测试数据
    ```
    {
     "price_orange":  2,
     "priority_apple":  13
    }
    ```
  - 加工规则
    ```
    e_set("op_min", op_min(ct_int(v("price_orange")),ct_int(v("priority_apple"))))
    ```


- 加工结果
```
price_orange: 2
priority_apple: 13
op_min: 2
```

10.2.2.14 事件检查函数

本文介绍事件检查函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
基本方法	e_has	判断日志字段是否存在。
	e_not_has	判断日志字段是否不存在。支持和其他函数组合使用。
表达式函数	e_search	提供一种简化，类似Lucene语法的事件搜索方式。支持和其他函数组合使用。
	e_match	判断当前日志字段的值是否满足正则表达式。支持和其他函数组合使用。
	e_match_any	判断当前日志字段的值是否满足正则表达式，任意字段匹配返回true，否则返回false。
	e_match_all	判断当前日志字段的值是否满足正则表达式，所有字段匹配返回true，否则返回false。

同时，事件检查函数可以与如下表达式函数配合使用：

类型	函数	说明
基本判断	op_and	逻辑and运算。
	op_or	逻辑or运算。
	op_not	逻辑not运算。
	op_nullif	判断两个表达式的取值。
	op_ifnull	返回第一个值不为None的表达式的值。
	op_coalesce	返回第一个值不为None的表达式的值。

e_has

判断字段是否存在。

- **函数格式**
```
e_has("key")
```
- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	日志的字段名。

- **返回结果**

  字段存在返回true，不存在返回false。

- **函数示例**

  判断日志是否存在content字段，存在则保留，不存在则丢弃。

  - 测试数据

    ```
    {
     "content": 123
    }
    ```

  - 加工规则

    ```
    e_keep(e_has("content"))
    ```

  - 加工结果

    ```
    content: 123
    ```

e_not_has

判断字段是否不存在。

- **函数格式**

  ```
  e_not_has("key")
  ```

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名称。

- **返回结果**

  字段不存在返回true，存在返回false。

- **函数示例**

  判断日志是否存在content字段，不存在则保留该日志，否则丢弃该日志。

  - 测试数据

    ```
    {
     "content": 123
    }
    ```

  - 加工规则

    ```
    e_if_else(e_not_has("content"),e_keep(),e_drop())
    ```

  - 加工结果

    日志被丢弃。

- **更多参考**

  支持和其他函数组合使用。

e_search

提供一种简化，类似Lucene语法的事件搜索方式。

说明

该功能在邀测期间，e_search函数处理日志流量比较小，请谨慎使用。

- **函数格式**

```
e_search(querystring)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
querystring	String	是	查询字符串，用于快速过滤日志的查询字符串。

- **返回结果**

  满足条件返回true，否则返回false。

- **函数参考示例**

```
# 全文
e_search("active error")# 全文：两个子串是OR关系，进行搜索。
e_search('"active error"')  # 全文：一个子串搜索。
# 字段:字符串
e_search("status: active")        # 单词搜索。
e_search('author: "john smith"')  # 带空格子串搜索。
e_search('field: active error')   # 相当于field:active OR "error"。
# 完全匹配
e_search('author== "john smith"')
# 通配符搜索，星号（*）匹配零个或多个字符，半角问号（?）匹配一个字符。
e_search("status: active*test")    # active*test中仅包含星号（*），可以不使用双引号（""）包裹。
e_search("status: active?good")    # active?good中仅包含半角问号（?），可以不使用双引号（""）包裹。
e_search("status== ac*tive?good") # 完全匹配。
# 搜索值转义，星号（*）或问号（?）需要使用反斜线（\）转义。
e_search('status: "\*\?()[]:="')  # \*\?()[]:=中包含特殊字符，需要使用双引号（""）包裹，除了星号（*）、半角问号（?）和反斜线（\）需要转义外，其他不用转义。
e_search("status: active\*test")  # active\*test中仅包含星号（*），可以不使用双引号（""）包裹。
e_search("status: active\?test")  # active\?test中仅包含半角问号（?），可以不使用双引号（""）包裹。
 # 字段名转义
e_search("\*\(1+1\)\?: abc")                  # 字段名不能用双引号（""）包裹，特殊字符用反斜线（\）转义。 e_search("__tag__\:__container_name__: abc")  # 用反斜线（\）转义。
e_search("中文字段: abc")                     # 直接写中文。
# 正则匹配
e_search('content~="正则表达式"')  # 正则匹配。
# 数字
e_search('count: [100, 200]')   # >=100 and <=200
e_search('count: [*, 200]')     # <=200
e_search('count: [200, *]')     # >=200
e_search('age >= 18')           # >= 18
e_search('age > 18')           # > 18
# 使用关系运算符
e_search("abc OR xyz")    # 关系运算符不区分大小写，OR和or效果一样。
e_search("abc and (xyz or zzz)")
e_search("abc and not (xyz and not zzz)")
e_search("abc && xyz")    # and
e_search("abc || xyz")    # or
e_search("abc || !xyz")   # or not
```

- 函数示例
  - 测试数据

    ```
    {
     "desc": "john smith is a player"
    }
    ```

  - 加工规则

    ```
    e_search('desc: "john smith"')
    ```


- 加工结果
  ```
  desc: john smith is a player
  desc: john smith is a player
  ```

- **更多参考**

  支持和其他函数组合使用。

e_match

判断当前日志字段的值是否满足正则表达式。

- **函数格式**
  ```
  e_match(key, regular_expression, full=true)
  ```

  说明

  e_match函数通常与op_not、op_and或者op_or结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么e_match("f1", ...)结果为false。
regular_expression	String	是	正则表达式。如果需要使用纯粹字符串匹配时（非正则表达式），可以使用函数str_regex_escape修饰正则表达式。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**

  返回字段匹配的判断结果true或false。

- **函数示例**

  判断字段k1的值是否为数字。

  - 测试数据
    ```
    {
     "k1": 123
    }
    ```
  - 加工规则
    ```
    e_set("e_match", e_match("k1",r'\d+'))
    ```
  - 加工结果
    ```
    k1: 123
    match: true
    ```

- **更多参考**

  支持和其他函数组合使用。

e_match_any

判断当前日志字段的值是否满足正则表达式，任意字段匹配返回true，否则返回false。

- **函数格式**
  ```
  e_match_any(key1, regular_expression1, key2, regular_expression2, ..., full=true)
  ```

  说明

  - 函数中key和regular_expression必须成对出现。
  - e_match_any函数通常与op_not、op_and或者op_or结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么e_match_any("f1", ...)结果为false。
regular_expression	String	是	正则模式。如果需要使用纯粹字符串匹配时（非正则），可以使用函数str_regex_escape修饰正则。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**

  返回字段匹配的判断结果true或false。

- **函数示例**

  e_match_any匹配，任意字段匹配则返回true。

  - 测试数据
    ```
    {
     "k1": 123,
     "k2": "abc",
     "k3": "abc123"
    }
    ```
  - 加工规则
    ```
    e_set("match",e_match_any('k1', r'\d+', 'k2', '.+'))
    ```
  - 加工结果
    ```
    k1:123
    k2:abc
    k3:abc123
    match:true
    ```

- **更多参考**

  支持和其他函数组合使用。

e_match_all

判断当前日志字段的值是否满足正则表达式，所有字段匹配返回true，否则返回false。

- **函数格式**
  ```
  e_match_all(key1, regular_expression1, key2, regular_expression2, ..., full=true)
  ```


说明

- 函数中key和regular_expression必须成对出现。
- e_match_all函数通常与op_not、op_and或者op_or结合使用。

- **参数说明**

参数名称	参数类型	是否必填	说明
字段名	String	是	字段名。当字段不存在时，视为当前子条件不匹配。例如：字段f1不存在，那么e_match_all("f1", ...)结果为false。
正则	String	是	正则模式。如果需要使用纯粹字符串匹配时（非正则），可以使用函数str_regex_escape修饰正则。
full	Bool	否	是否完全匹配，默认为true表示完全匹配。

- **返回结果**

  返回字段匹配的判断结果true或false。

- **函数示例**
  - 测试数据

    ```
    {
     "k1": 123,
     "k2": "abc",
     "k3": "abc123"
    }
    ```

  - 加工规则

    ```
    e_set("match", e_match_all("k1", r"\d+", "k2", r"\d+"))
    ```

  - 加工结果

    ```
    k1:123
    k2:abc
    k3:abc123
    match:false
    ```

  - **更多参考**

    支持和其他函数组合使用。

10.2.3 全局操作函数

10.2.3.1 映射富化函数

本文介绍映射富化函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
字段映射	dict_map	与目标数据字典进行映射，根据输入的字段映射一个新字段。支持和其他函数组合使用。

	e_table_map	与目标表格进行映射，根据输入的字段名称返回字段值。支持和其他函数组合使用。
搜索映射	e_search_dict_map	对关键字（查询字符串）以及其匹配的值的字典数据进行映射。支持和其他函数组合使用。
	e_search_table_map	对某列（查询字符串）以及其匹配的值的表格数据进行映射。

dict_map

与目标数据字典进行映射，根据输入的字段映射一个新字段。

- **函数格式**
  ```
  e_dict_map(data, field, output_field, case_insensitive=true, missing=None, mode="overwrite")
  ```
- **参数说明**

参数名称	数据类型	是否必填	说明
data	Dict	是	目标数据字典。必须为标准的{key01:value01,key01:value02,...}格式，且必须是字符串。例如{"1": "TCP", "2": "UDP", "3": "HTTP", "*": "Unknown"}。
field	String或者String List	是	一个字段名或者多个字段名的列表。多个字段时： • 依次对匹配到的值进行映射。 • 如果匹配命中多条日志，且mode的取值为overwrite时，则最后一个会覆盖前面的结果。 • 当没有匹配到任何字段，则使用missing参数的值作为匹配值。
output_field	String	是	输出字段的名称。
case_insensitive	Boolean	否	匹配时大小写是否不敏感。 • true（默认值）：不敏感。 • false：敏感 **说明** 如果字典中存在同一个关键字的不同大小写，且case_insensitive为true时，会优先选择大小写完全匹配的值。如果没有，则随机选择一个。

missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_field。默认为None表示不做映射赋值操作。 **说明** 如果字典中包含匹配星号（*），由于星号（*）的优先级高于missing，此时missing参数不生效。
mode	String	否	字段的覆盖模式。默认为overwrite。 取值为：fill，fill-auto,add,add-auto,overwrite,overwrite-auto

- **返回结果**

  返回附带新字段的日志。

- **函数示例**

  a. 示例1：根据测试数据中pro字段的值和目标数据字典，输出新字段protocol。

  - 测试数据

    ```
    {
     "data":  123,
     "pro":  1
    }
    ```

  - 加工规则

    ```
    e_dict_map(
       {"1": "TCP", "2": "UDP", "3": "HTTP", "6": "HTTPS", "*": "Unknown"},
       "pro",
       "protocol",
    )
    ```

  - 加工结果

    ```
    data:  123
    pro:  1
    protocol:  TCP
    ```

  b. 示例2：根据测试数据中status字段的值和目标数据字典，输出新字段message。

  - 测试数据（三条测试日志）

    ```
    {
      "status":[500]
    }
    {
      "status":[400]
    }
    {
      "status":[200]
    }
    ```

  - 加工规则

    ```
    e_dict_map({"400": "错误", "200": "正常", "*": "其他"}, "status", "message")
    ```

  - 加工结果

    ```
    status:  500
    message:  其他
    status:  400
    message:  错误
    ```



```
status: 200
message: 正常
```

- **更多参考**

  支持和其他函数组合使用。

e_table_map

与目标表格进行映射，根据输入的字段名称返回字段值。

- **函数格式**

```
e_table_map(data, field, output_fields, missing=None, mode="fill-auto")
```

- **参数说明**

参数名称	数据类型	是否必填	说明
data	Table	是	目标表格。 **说明** 如果采用资源函数res_rds_mysql和res_log_logstore_pull作为数据源，则为了提升处理性能，请尽量设置primary_keys参数。
field	String、String List或Tuple List	是	日志中映射到表格的源字段。如果日志中不存在对应字段，则不进行任何操作。
output_fields	String、String List或Tuple List	是	映射后的字段。例如["province", "pop"]。
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_fields。默认为None表示不做映射赋值操作。如果目标字段是多列，则missing可以是一个长度与目标字段数一致的默认值列表。 **说明** 如果表格中包含匹配星号（*），由于星号（*）的优先级高于missing，此时missing参数将不起作用。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**

  返回附带新字段值的日志。

- **函数示例**

  a. 示例1：在映射表格中查找对应行，根据city字段返回province字段的值。

  - 测试数据

```
{
 "data": 123,
 "city": "nj"
}
```


- 加工规则
```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"), "city", "province"
)
```
- 加工结果
```
data: 123
city: nj
province: js
```

b. 示例2：在映射表格中查找对应行，根据city字段返回province字段和pop字段的值。

- 测试数据
```
{
 "data": 123,
 "city": "nj"
}
```
- 加工规则
```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
    "city",
    ["province", "pop"],
)
```
- 加工结果
```
data: 123
city: nj
province: js
pop: 800
```

c. 示例3：使用tab_parse_csv函数构建映射表格，根据city字段返回province字段和pop字段的值。

- 测试数据
```
{
 "data": 123,
 "city": "nj"
}
```
- 加工规则
```
e_table_map(
    tab_parse_csv("city#pop#province\nnj#800#js\nsh#2000#sh", sep="#"),
    "city",
    ["province", "pop"],
)
```
- 加工结果
```
data: 123
city: nj
province: js
pop: 800
```

d. 示例4：使用tab_parse_csv函数构建映射表格，根据city字段返回province字段和pop字段的值。

- 测试数据
```
{
 "data": 123,
 "city": "nj"
}
```
- 加工规则
```
e_table_map(
    tab_parse_csv(
```



```
        "city,pop,province\n|nj|,|800|,|js|\n|shang hai|,2000,|SHANG,HAI|", quote="|"
    ),
    "city",
    ["province", "pop"],
)
```

- 加工结果

```
data: 123
city: nj
province: js
pop: 800
```

e. 示例5：日志匹配字段与映射表格中字段不一样。在映射表格中查找对应行，根据cty或city字段返回province字段的值。

- 测试数据

```
{
 "data": 123,
 "city": "nj"
}
```

- 加工规则

```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
    [("city", "city")],
    "province"
)
```

- 加工结果

```
data: 123
city: nj
province: js
```

f. 示例6：日志匹配字段与映射表格中字段不一样，并且对输出字段进行重命名。

- 测试数据

```
{
 "data": 123,
 "city": "nj"
}
```

- 加工规则

```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
    [("city", "city")],
    [("province", "pro")],
)
```

- 加工结果

```
data: 123
city: nj
pro: js
```

g. 示例7：多个日志匹配字段。

- 测试数据

```
{
 "data": 123,
 "city": "nj",
 "pop": 800
}
```

- 加工规则

```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
    ["city", "pop"],
```



```
    "province",
)
```

- 加工结果

```
data: 123
city: nj
pop: 800
province: js
```

h. 示例8：多个日志匹配字段，且日志匹配字段与映射表格字段不一样。

- 测试数据

```
{
 "data": 123,
 "city": "nj",
 "pp": 800
}
```

- 加工规则

```
e_table_map(
    tab_parse_csv("city,pop,province\nnj,800,js\nsh,2000,sh"),
    [("city", "city"), ("pp", "pop")],
    "province",
)
```

- 加工结果

```
data: 123
city: nj
pp: 800
province: js
```

- **更多参考**

  支持和其他函数组合使用。

e_search_dict_map

对关键字（查询字符串）以及其匹配的值的字典数据进行映射。

- **函数格式**

```
e_search_dict_map(data, output_field, multi_match=false, multi_join=" ", missing=None, mode="overwrite")
```

- **参数说明**

参数名称	数据类型	是否必填	说明
data	Dict	是	映射关系的字典。.必须为标准的{key01:value01,key01:value02,...}格式，且关键字key必须是查询字符串。
output_field	String	是	输出字段的名称。
multi_match	Boolean	否	是否允许匹配多个字段。默认为false表示不匹配多个字段，会返回匹配到的最后一个字段值。支持使用multi_join拼接多个匹配的值。
multi_join	String	否	匹配多个字段时，多值的连接字符串，默认为空格。当multi_match值为true时生效。

missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_field。默认为None表示不做映射赋值操作。 **说明** 如果字典中包含默认匹配星号（*），由于星号（*）的优先级高于missing，此时missing将不起作用。
mode	String	否	字段的覆盖模式。默认为overwrite。

- **返回结果**
  返回查询匹配中后的映射结果。
- **函数示例**
  a. 示例1：匹配模式。
     - 测试数据
       ```
       {
        "data":123 ,
        "pro":1
       }
       ```
     - 加工规则
       ```
       e_search_dict_map ({"pro==1": "TCP", "pro==2": "UDP", "pro==3": "HTTP"}, "protocol")
       ```
     - 加工结果
       ```
       data:123
       pro:1
       protocol:TCP
       ```

  b. 示例2：根据字段值的不同开头进行映射。
     - 测试数据
       ```
       {
        "status":"200,300"
       }
       ```
     - 加工规则
       ```
       e_search_dict_map(
          {
             "status:2??": "ok",
             "status:3??": "redirect",
              "status:4??": "auth",
              "status:5??": "server_error",
          },
          "status_desc",
         multi_match=true,
         multi_join="测试",
       )
       ```
     - 加工结果
       ```
       status:200,300
       status_desc:ok测试redirect
       ```
- 更多参考
  支持和其他函数组合使用。

e_search_table_map

对某列（查询字符串）以及其匹配的值的表格数据进行映射。

- **函数格式**
```
e_search_table_map(data, inpt, output_fields, multi_match=false, multi_join=" ", missing=None, mode="fill-auto")
```
- **参数说明**

参数名称	数据类型	是否必填	说明
data	Table	是	映射关系的表格，表格某一列必须是查询字符串。
inpt	String	是	表格中用于匹配搜索的字段名。
output_fields	String，String List或Tuple List	是	表格中映射出的字段，可以是字符串、列表或者其名称映射元组的列表。
multi_match	Boolean	否	是否允许匹配多个字段。默认为false表示不匹配多个字段，会返回匹配到的第一个字段值。支持使用multi_join来拼接多个匹配的值。
multi_join	String	否	匹配多个字段时，多值的连接字符串，默认为空格。当multi_match值为true时生效。
missing	String	否	无匹配字段时，将该参数的取值赋给输出字段output_fields。默认为None表示不做映射赋值操作。 **说明** 如果表格中包含默认匹配*，由于*的优先级高于missing，此时missing将不起作用。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**

  返回查询匹配中后的映射结果。
- **函数示例**

  a. 示例1：根据映射关系的表格，将日志中city字段映射出pop和province字段。

  - 测试数据
```
{
 "data": 123,
 "city": "sh"
}
```
    例如，以下映射关系的表格，其中search列是查询字符串。

search	pop	province
city==nj	800	js

city==sh	2000	sh

- 加工规则

```
e_search_table_map(
    tab_parse_csv("search,pop,province\ncity==nj,800,js\ncity==sh,2000,sh"),
    "search",
    ["pop", "province"],
)
```

- 加工结果

```
data: 123
city: sh
province: sh
pop: 2000
```

b. 示例2：overwrite模式。

- 测试数据

```
{
 "data": 123,
 "city": "nj",
 "province":""
}
```

- 加工规则

```
e_search_table_map(
   tab_parse_csv("search,pop,province\ncity==nj,800,js\ncity==sh,2000,sh"),
    "search",
    "province",
    mode="overwrite",
)
```

- 加工结果

```
pop: 800
data: 123
city: nj
province: js
```

c. 示例3：无匹配时目标字段的值由missing指定。

- 测试数据

```
{
 "data": 123,
 "city": "wh",
 "province":""
}
```

- 加工规则

```
e_search_table_map(
    tab_parse_csv("search,pop,province\ncity==nj,800,\ncity==sh,2000,sh"),
    "search",
    "province",
    missing="Unknown",
)
```

- 加工结果

```
data: 123
city: wh
province: Unknown
```

d. 示例4：允许匹配多个字段（multi_match模式）。

- 测试数据

```
{
 "data": 123,
```



```
"city": "nj,sh",
"province":""
}
```

- 加工规则

```
e_search_table_map(
    tab_parse_csv("search,pop,province\ncity:nj,800,js\ncity:sh,2000,sh"),
    "search",
    "province",
    multi_match=true,
    multi_join=",",
)
```

- 加工结果

```
data: 123
city: nj,sh
province: js,sh
```

10.2.3.2 字段值提取函数

本文介绍字段值提取函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
正则提取	e_regex	根据正则表达式提取字段的值并赋值给其他字段。支持和其他函数组合使用。
JSON提取	e_json	对特定字段中的JSON对象进行JSON操作，包括JSON展开、JMES提取或者JMES提取后再展开。支持和其他函数组合使用。
分隔符提取	e_csv、e_psv、e_tsv	使用自定义的分隔符与预定义的字段名，从特定字段中提取多个字段。 ● e_csv：默认分隔符为半角逗号（,）。 ● e_psv：默认分隔符为竖线（|）。 ● e_tsv：默认分隔符为\t。 支持和其他函数组合使用。
KV模式提取	e_kv	通过quote提取多个源字段中的键值对信息。支持和其他函数组合使用。
	e_kv_delimit	通过分隔符提取源字段中的键值对信息。

e_regex

根据正则表达式提取字段的值并赋值给其他字段。

- **函数格式**

```
e_regex(key,正则表达式,fields_info,mode="fill-auto",pack_json=None)
```

- **参数说明**

参数名称	参数类型	是否必填	说明
key	任意	是	源字段名。如果字段不存在，则不进行任何操作。特殊字段名的设置请参见事件类型。
正则表达式	String	是	提取字段的正则表达式。 支持捕获组和非捕获组正则表达式。 **说明** 非捕获有时需要使用分组，且使用前缀?:。例如\w+@\w+\.\w(?:\.\cn)?。关于分组不捕获请参见非捕获组。
fields_info	String/ List/ Dict	否	匹配后目标字段名。当正则表达式参数没有配置命名捕获的名称时，必须配置该参数。
mode	String	否	字段的覆盖模式。默认为fill-auto。关于更多字段值及含义请参见字段提取检查与覆盖模式。
pack_json	String	否	将正则表达式的所有匹配结果打包放入pack_json指定的字段中。默认值是None，表示不打包。

- **返回结果**

  返回附带新字段值的日志。

- **函数示例**

  a. 示例1：提取字段中符合表达式的值。

  - 测试数据

    ```
    {
     "msg": "192.168.0.1 http://... 127.0.0.0"
    }
    ```

  - 加工规则

    ```
    # 提取字段msg中出现的第一个IP地址。
    e_regex("msg",r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}","ip")
    ```

  - 加工结果

    ```
    msg: 192.168.0.1 http://... 127.0.0.0
    ip: 192.168.0.1
    ```

  b. 示例2：提取字段中符合表达式的多个值。

  - 测试数据

    ```
    {
     "msg": "192.168.0.1 http://... 127.0.0.0"
    }
    ```

  - 加工规则

    ```
    # 提取字段msg中出现的两个IP地址，分别赋值给server_ip和client_ip。
    e_regex("msg",r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}",["server_ip","client_ip"])
    ```

  - 加工结果

    ```
    msg: 192.168.0.1 http://... 127.0.0.0
    server_ip: 192.168.0.1
    client_ip: 127.0.0.0
    ```


c. 示例3：通过捕获组提取符合表达式的值。

- 测试数据

```
{
 "content": "start sys version: deficience, err: 2"
}
```

- 加工规则

```
# 使用正则表达式捕获content中的version和error值。
e_regex("content",r"start sys version: (\w+),\s*err: (\d+)",["version","error"])
```

- 加工结果

```
content: start sys version: deficience, err: 2
error: 2
version: deficience
```

d. 示例4：通过命名捕获组提取字段值。

- 测试数据

```
{
 "content": "start sys version: deficience, err: 2"
}
```

- 加工规则

```
e_regex("content",r"start sys version: (?P<version>\w+),\s*err: (?P<error>\d+)")
```

- 加工结果

```
content:  start sys version: deficience, err: 2
error:  2
version:  deficience
```

e. 示例5：使用正则表达式捕获字段dict中的值，并动态命名字段名和其值。

- 测试数据

```
{
 "dict": "verify:123"
}
```

- 加工规则

```
e_regex("dict",r"(\w+):(\d+)",{r"k_\1": r"v_\2"})
```

- 加工结果

```
dict: verify:123
k_verify: v_123
```

f. 示例6：提取字段中符合表达式的值，并打包赋值给name字段。

- 测试数据

```
{
 "msg": "192.168.0.1 http://... 127.0.0.0"
}
```

- 加工规则

```
e_regex("msg", r"\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}", "ip", pack_json="name")
```

- 加工结果

```
msg:192.168.0.1 http://... 127.0.0.0
name:{"ip": "192.168.0.1"}
```

g. 示例7：使用正则表达式提取字段dict中的值，动态命名字段名和其值，并打包赋值给name字段。

- 测试数据

```
{
 "dict": "x:123, y:456, z:789"
}
```


- 加工规则
  ```
  e_regex("dict", r"(\w+):(\d+)", {r"k_\1": r"v_\2"}, pack_json="name")
  ```
- 加工结果
  ```
  dict:x:123, y:456, z:789
  name:{"k_x": "v_123", "k_y": "v_456", "k_z": "v_789"}
  ```

h. 示例8：通过捕获组提取符合表达式的值，并打包赋值给name字段。

- 测试数据
  ```
  {
   "content": "start sys version: deficience, err: 2"
  }
  ```
- 加工规则
  ```
  e_regex( "content", r"start sys version: (\w+),\s*err: (\d+)", ["version",
  "error"],pack_json="name")
  ```
- 加工结果
  ```
  content:start sys version: deficience, err: 2
  name:{"version": "deficience", "error": "2"}
  ```

- **更多参考**

  支持和其他函数组合使用。

e_json

对特定字段中的JSON对象进行JSON操作，包括JSON展开、JMES提取或者JMES提取后再展开。

- **函数格式**
  ```
  e_json(key, expand=None, depth=100, prefix="__", suffix="__", fmt="simple", sep=".",
        expand_array=true, fmt_array="{parent}_{index}",
        include_node=r"[\u4e00-\u9fa5\u0800-\u4e00a-zA-Z][\w\-\.]*",
         exclude_node="", include_path="", exclude_path="",
       jmes="", output="", jmes_ignore_none=false, mode='fill-auto'
  )
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
key	String	是	源字段名。如果字段不存在，则不进行任何操作。
expand	Boolean	否	是否将字段展开。 • 没有配置jmes参数时，则默认为true，表示展开。 • 配置jmes参数时，则默认为false，表示不展开。
depth	Number	否	字段展开的深度。取值范围为1~2000，1表示只展开第一层，默认为100层。
prefix	String	否	展开时添加为字段名的前缀。
suffix	String	否	展开时添加为字段名的后缀。

fmt	String	否	格式化方式。取值： ● simple（默认值）：表示将节点名作为字段名。展示形式为{prefix}{current}{suffix}。 ● full：表示将父节点与当前节点合并作为字段名。展示形式为{parent_list_str}{sep}{prefix}{current}{suffix}。分隔符是由sep参数指定，默认为.。 ● parent：表示用完整的路径作为字段名。展示形式为{parent}{sep}{prefix}{current}{suffix}。分隔符是由sep参数指定，默认为.。 ● root：表示将根节点与当前节点合并作为字段名。展示形式为{parent_list[0]}{sep}{prefix}{current}{suffix}。分隔符由sep参数指定，默认为.。
sep	String	否	父子节点格式化的分隔符。当fmt取值为full、parent或root时需要设置。默认为.。
expand_array	Boolean	否	是否将数组展开。默认为true表示展开数组。
fmt_array	String	否	数组展开的格式化方式，格式为{parent_rlist[0]}_{index}。也可以使用最多五个占位符自定义格式化字符串：parent_list，current，sep，prefix，suffix。
include_node	String/Number	否	节点允许名单，表示过滤时包含的节点名称。默认只有中文、数字、字母和_.-的节点才会被自动展开。
exclude_node	String	否	节点限制名单，表示过滤时排除的节点名称。
include_path	String	否	节点允许名单，表示过滤时包含的节点路径。
exclude_path	String	否	节点限制名单，表示过滤时排除的节点路径。
jmes	String	否	将字段值转化为JSON对象并通过JMES提取特定值。
output	String	否	通过JMES提取特定值时输出的字段名。
jmes_ignore_none	Boolean	否	当JMES提取不到值时是否忽略。默认为true表示忽略，否则输出一个空字符串。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- JSON展开过滤
  - 如果设置了节点允许名单，则内容必须包含在节点允许名单中然后才会在结果中出现。节点允许名单正则示例：e_json("json_data_filed", ..., include_node=r'key\d+')。
  - 如果设置了节点限制名单，则内容必须包含在节点限制名单中然后才不会在结果中出现。节点限制名单正则示例：e_json("json_data_filed", ..., exclude_node=r'key\d+')。
  - 展开节点路径：正则include_path 与 exclue_path从路径开头匹配，匹配路径是以.分隔。
- JMES过滤

  使用JMES选择、计算。
  - 选择特定JSON路径下的元素属性列表：e_json(..., jmes="cve.vendors[*].product",output="product")
  - 用逗号拼接特定JSON路径下的元素属性：e_json(..., jmes="join(',', cve.vendors[*].name)",output="vendors")
  - 计算特定JSON路径下元素的最大属性值：e_json(..., jmes="max(words[*].score)",output="hot_word")
  - 当特定路径不存在或为空时，返回一个空字符串：e_json(..., jmes="max(words[*].score)",output="hot_word", jmes_ignore_none=false)
- parent_list和parent_rlist，以如下示例说明。

  测试数据：

  ```
  {
   "data": { "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
  }
  ```

  parent_list是将父节点从左到右排列。

  ```
  e_json("data", fmt='{parent_list[0]}-{parent_list[1]}#{current}')
  ```

  得到的日志：

  ```
  data:{ "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
  data-k2#k3:200
  data-k2#k5:300
  ```

  parent_rlist是将父节点从右到左排列。

  ```
  e_json("data", fmt='{parent_rlist[0]}-{parent_rlist[1]}#{current}')
  ```

  得到的日志：

  ```
  data:{ "k1": 100,"k2": {"k3": 200,"k4": {"k5": 300}}}
  k2-data#k3:200
  k4-k2#k5:300
  ```

- 返回结果

  返回附带新字段值的日志。
- 函数示例

  a. 示例1：字段展开操作。
  - 测试数据

    ```
    {
     "data": {"k1": 100, "k2": 200}
    }
    ```


- 加工规则
```
e_json("data",depth=1)
```
- 加工结果
```
data: {"k1": 100, "k2": 200}
k1: 100
k2: 200
```

b. 示例2：给字段名添加前缀和后缀。

- 测试数据
```
{
 "data": {"k1": 100, "k2": 200}
}
```
- 加工规则
```
e_json("data", prefix="data_", suffix="_end")
```
- 加工结果
```
data: {"k1": 100, "k2": 200}
data_k1_end: 100
data_k2_end: 200
```

c. 示例3：将字段按照不同格式展开。

- 测试数据
```
{
 "data": {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
}
```
- fmt=full格式
```
e_json("data", fmt='full')
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
data.k1: 100
data.k2.k3: 200
data.k2.k4.k5: 300
```
- fmt=parent格式
```
e_json("data", fmt='parent')
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
data.k1: 100
k2.k3: 200
k4.k5: 3000
```
- fmt=root格式
```
e_json("data", fmt='root')
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
data.k1: 100
data.k3: 200
data.k5: 300
```

d. 示例4：使用指定分隔符、字段名前缀和字段名后缀提取JSON

- 测试数据
```
{
 "data": {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
}
```
- 加工规则
```
e_json("data", fmt='parent', sep="@", prefix="__", suffix="__")
```
- 加工结果
```
data: {"k1": 100, "k2": {"k3": 200, "k4": {"k5": 300} } }
data@__k1__:  100
k2@__k3__:  200
k4@__k5__:  300
```


e. 示例5：指定fmt_array参数，按照数组方式提取JSON。

- 测试数据

```
{
 "people": [{"name": "xm", "sex": "boy"}, {"name": "xz", "sex": "boy"}, {"name": "xt", "sex": "girl"}]
}
```

- 加工规则

```
e_json("people", fmt='parent', fmt_array="{parent_rlist[0]}-{index}")
```

- 加工结果

```
people: [{"name": "xm", "sex": "boy"}, {"name": "xz", "sex": "boy"}, {"name": "xt", "sex": "girl"}]
people-0.name: xm
people-0.sex: boy
people-1.name: xz
people-1.sex: boy
people-2.name: xt
people-2.sex: girl
```

f. 示例6：使用JMES提取JSON对象。

- 测试数据

```
{
 "data": { "people": [{"first": "James", "last": "d"},{"first": "Jacob", "last": "e"}],"foo": {"bar": "baz"}}
}
```

- 加工规则

```
e_json("data", jmes='foo', output='jmes_output0')
e_json("data", jmes='foo.bar', output='jmes_output1')
e_json("data", jmes='people[0].last', output='jmes_output2')
e_json("data", jmes='people[*].first', output='jmes_output3')
```

- 加工结果

```
data: { "people": [{"first": "James", "last": "d"},{"first": "Jacob", "last": "e"}],"foo": {"bar": "baz"}}
jmes_output0: {"bar": "baz"}
jmes_output1: baz
jmes_output2: d
jmes_output3: ["james", "jacob"]
```

- **更多参考**

  支持和其他函数组合使用。

e_csv、e_psv、e_tsv

使用自定义的分隔符与预定义的字段名，从特定字段中提取多个字段。

- e_csv：默认分隔符为半角逗号（,）。
- e_psv：默认分隔符为竖线（|）。
- e_tsv：默认分隔符为\t。
- **函数格式**

```
e_csv(源字段名, 目标字段列表, sep=",", quote='"', restrict=true, mode="fill-auto")
e_psv(源字段名, 目标字段列表, sep="|", quote='"', restrict=true, mode="fill-auto")
e_tsv(源字段名, 目标字段列表, sep="\t", quote='"', restrict=true, mode="fill-auto")
```

- **参数说明**

参数名称	参数类型	是否必填	说明

源字段名	任意	是	源字段名。如果字段不存在，则不进行任何操作。
目标字段列表	任意	是	字段值经过分隔符分隔后的每个值对应的字段名。可以是字符串的列表，例如：["error", "message", "result"]。 当字段名中不包含逗号时，也可以直接用逗号作为分隔字符，例如："error, message, result"。
sep	String	否	分隔符，只能是单个字符。
quote	String	否	引用符，用于包裹值的字符。当值包含分隔符时需要使用。
restrict	Boolean	否	是否采用严格模式，默认为false表示非严格模式。当分隔值的个数与目标字段列表数不一致时： ● 严格模式下不进行任何操作。 ● 非严格模式下对前几个可以配对的字段进行赋值。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**
  返回附带新字段值的日志。
- **函数示例**
  以e_csv为例，e_psv和e_tsv功能类似。
  - 测试数据
    ```
    {
     "content": "192.168.0.100,10/Jun/2019:11:32:16 +0800,example.aadoc.com,GET /zf/11874.html HTTP/1.1,200,0.077,6404,192.168.0.100:8001,200,0.060,https://image.developer.aadoc.com/s?q=%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A%E6%B3%95&from=wy878378&uc_param_str=dnntnwvepffrgibijbprsvdsei,-,Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebKit/537.36,-,-"
    }
    ```
  - 加工规则
    ```
    e_csv("content", "remote_addr, time_local,host,request,status,request_time,body_bytes_sent,upstream_addr,upstream_status, upstream_response_time,http_referer,http_x_forwarded_for,http_user_agent,session_id,guid")
    ```
  - 加工结果
    ```
    content:  192.168.0.100,10/Jun/2019:11:32:16 +0800,example.aadoc.com,GET /zf/11874.html HTTP/1.1,200,0.077,6404,192.168.0.100:8001,200,0.060,https://image.developer.aadoc.com/s?q=%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A%E6%B3%95&from=wy878378&uc_param_str=dnntnwvepffrgibijbprsvdsei,-,Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Mobile Safari/537.36,-,-
     body_bytes_sent:  6404
    guid:  -
    host:  example.aadoc.com
    http_referer:  https://image.developer.aadoc.com/s?q=%E8%9B%8B%E8%8A%B1%E9%BE%99%E9%A1%BB%E9%9D%A2%E7%9A%84%E5%81%9A%E6%B3%95&from=wy878378&uc_param_str=dnntnwvepffrgibijbprsvdsei
    http_user_agent:  Mozilla/5.0 (Linux; Android 9; HWI-AL00 Build/HUAWEIHWI-AL00) AppleWebKit/537.36
    http_x_forwarded_for:  -
    remote_addr:  192.168.0.100
    ```



```
request:  GET /zf/11874.html HTTP/1.1
request_time:  0.077
session_id:  -
status:  200
time_local:  10/Jun/2019:11:32:16 +0800
topic:  syslog-forwarder
upstream_addr:  192.168.0.100:800
1upstream_response_time:  0.060
upstream_status:  200
```

- **更多参考**

  支持和其他函数组合使用。

e_kv

通过quote提取多个源字段中的键值对信息。

- **函数格式**

```
e_kv(源字段或源字段列表, sep="=", quote="", escape=false, prefix="", suffix="", mode="fill-auto")
```

- **参数说明**

参数名称	参数类型	是否必填	说明
源字段或源字段列表	字符串或字符串列表	是	字段名或多个字段名的列表。
sep	String	否	关键字与值的正则表达式的分隔符串，默认为=，不限于单个字符。 **说明** 可以使用非捕获分组，但不能使用捕获分组。
quote	String	否	引用符，用于包裹值的字符。默认为"。 **说明** 提取的动态键值对的值一般需要quote来包括，例如：a="abc"，b="xyz"如果提取对象中不包含，则只提取如下字符集的值：中文字母数字_-.%~。例如a=中文ab12_-.%~|abc b=123可以提取a: 中文ab12_-.%~，b: 123。
escape	Boolean	否	是否自动提取反转字符的值。默认为false表示否。例如key="abc\"xyz"默认提取字段key的值为abc\，设置escape=true时，提取的值为abc"xyz。
prefix	String	否	给提取的字段名添加前缀。
suffix	String	否	给提取的字段名添加后缀。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**

  返回附带新字段值的日志。

- **函数示例**

a. 示例1：使用默认分隔符=提取键值对信息。

- 测试数据

```
{
 "http_refer": "https://video.developer.aadoc.com/s?q=asd&a=1&b=2"
}
```

> **说明**
>
> 如果测试数据为request_uri: a1=1&a2=&a3=3，a2值为空，则使用e_kv()函数无法提取出a2。您可以使用e_regex()函数进行提取，例如e_regex("request_uri",r'(\w+)=([^=&]*)',{r"\1":r"\2"},mode="overwrite")。

- 加工规则

```
e_kv("http_refer")
```

- 加工结果

```
http_refer: https://video.developer.aadoc.com/s?q=asd&a=1&b=2
q: asd
a: 1
b: 2
```

b. 示例2：给字段名增加前缀和后缀。

- 测试数据

```
{
 "http_refer": "https://video.developer.aadoc.com/s?q=asd&a=1&b=2"
}
```

- 加工规则

```
e_kv(
    "http_refer",
    sep="=",
    quote='"',
    escape=false,
    prefix="data_",
    suffix="_end",
    mode="fill-auto",
)
```

- 加工结果

```
http_refer: https://video.developer.aadoc.com/s?q=asd&a=1&b=2
data_q_end: asd
data_a_end: 1
data_b_end: 2
```

c. 示例3：提取字段content2中的键值对信息，使用escape参数提取反转字符的值。

- 测试数据

```
{
 "content2": "k1:\"v1\\"abc\", k2:\"v2\", k3: \"v3\""
}
```

- 加工规则

```
e_kv("content2", sep=":", escape=true)
```

- 加工结果

```
content2:  k1:"v1\"abc", k2:"v2", k3: "v3"
k1: v1"abc
k2: v2
k3: v3
```

- **更多参考**

  支持和其他函数组合使用。

e_kv_delimit

通过分隔符提取源字段中的键值对信息。

- **函数格式**
  ```
  e_kv_delimit(源字段或源字段列表, pair_sep=r"\s", kv_sep="=", prefix="", suffix="", mode="fill-auto")
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
源字段或源字段列表	字符串或字符串列表	是	字段名或多个字段名的列表。
pair_sep	String	否	用于分隔键值对的正则字符集，默认为\s。例如\s\w、abc\s等。 **说明** 如果您需要使用字符串对字段进行分隔，推荐您使用str_replace或regex_replace将字符串转换成字符作为分隔符，然后再使用e_kv_delimit函数对字段进行分隔。
kv_sep	String	否	用于分隔键值对的正则字符串，默认为=，不限于单个字符。 **说明** 可以使用非捕获分组，但不能使用捕获分组。
prefix	String	否	给提取的字段名添加前缀。
suffix	String	否	给提取的字段名添加后缀。
mode	String	否	字段的覆盖模式。默认为fill-auto。

- **返回结果**
  返回附带新字段值的日志。
- **函数示例**
  a. 示例1：使用默认分隔符=提取键值对信息。
     - 测试数据
       ```
       {
        "data": "i=c1 k1=v1 k2=v2 k3=v3"
       }
       ```
       说明
       如果测试数据为request_uri: a1=1&a2=&a3=3，a2值为空，则使用e_kv_delimit()函数无法提取出a2。您可以使用e_regex()函数进行提取，例如e_regex("request_uri",r'(\w+)=([^=&]*)',{r"\1":r"\2"}, mode="overwrite")。
     - 加工规则
       ```
       e_kv_delimit("data")
       ```
     - 加工结果
       ```
       data: i=c1 k1=v1 k2=v2 k3=v3
       i: c1
       k2: v2
       ```



```
k1: v1
k3: v3
```

b. 示例2：使用分隔符&?提取键值对信息。

- 测试数据

```
{
 "data": "k1=v1&k2=v2?k3=v3"
}
```

- 加工规则

```
e_kv_delimit("data",pair_sep=r"&?")
```

- 加工结果

```
data: k1=v1&k2=v2?k3=v3
k2: v2
k1: v1
k3: v3
```

c. 示例3：使用正则表达式提取键值对信息。

- 测试数据

```
{
 "data": "k1=v1 k2:v2 k3=v3"
}
```

- 加工规则

```
e_kv_delimit("data", kv_sep=r"(?:=|:)")
```

- 加工结果

```
data: k1=v1 k2:v2 k3=v3
k2: v2
k1: v1
k3: v3
```

10.2.3.3 字段操作函数

本文介绍字段操作函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
V	获得日志特定字段的值。当同时传入多个字段名时，返回日志中第一个存在的字段的值。支持和其他函数组合使用。
e_set	添加新字段或为现有字段设置新的字段值。支持和其他函数组合使用。
e_drop_fields	删除符合条件的日志字段。支持和其他函数组合使用。
e_keep_fields	保留符合条件的日志字段。
e_pack_fields	打包日志字段，并输出到新的字段中。
e_rename	重命名符合条件的日志字段名称。支持和其他函数组合使用。

v

调用v函数获得日志特定字段的值。当同时传入多个字段名时，返回日志中第一个存在的字段的值。

- **函数格式**
  ```
  v(key, ..., default=None)
  ```
- **参数说明**

参数	参数类型	是否必填	说明
key	String	是	指定字段名。
default	任意	否	指定的字段名不存在时，返回default的值。默认值为None。

- **返回结果**

  返回日志中第一个存在的字段值。不存在时返回default参数的值。
- **函数示例**

  将content字段的值赋给test_content字段。
  - 测试数据
    ```
    {
     "content": "hello"
    }
    ```
  - 加工规则
    ```
    e_set("test_content", v("content"))
    ```
  - 加工结果
    ```
    content: hello
    test_content: hello
    ```
- **更多参考**

  支持和其他函数组合使用。

e_set

调用e_set函数添加新字段或为现有字段设置新的字段值。

- **函数格式**
  ```
  e_set(key1, value1, key2, value2, mode="overwrite")
  ```

  说明

  - 函数中key1和value1必须成对出现。
  - 通过e_set函数设置时间字段F_TIME或__time__时，必须设置为数字、字符串。

  ```
  e_set(F_TIME, "abc")  # 错误
  e_set(F_TIME, "12345678")  # 正确
  ```
- **参数说明**

参数	参数类型	是否必填	说明
key	String	是	目标字段名，也可以通过字符串表达式获得该字段名。

value	任意	是	新的字段值。非字符串都转化成字符串放入日志中，其中元组、列表、字典会转换成JSON对象的字符串。 **说明** 如果传递的值是None，则不会进行更新操作。
mode	String	否	字段的覆盖模式。默认为overwrite。

- **返回结果**

  返回更新后的日志。

- **函数示例**

  a. 示例1：为字段设置固定值。

  添加一个新字段city，字段值为上海。

  ```
  e_set("city", "上海")
  ```

  b. 示例2：复制字段值。

  调用单个表达式函数，将现有字段ret的值，赋给新字段result。

  - 原始数据

    ```
    {"ret": "value"}
    ```

  - 加工规则

    ```
    e_set("result", v("ret"))
    ```

  - 加工结果

    ```
    ret：value
    result：value
    ```

  c. 示例3：动态设置值。

  调用组合表达式函数，获取第一个存在的字段值，返回其小写格式并赋值给字段result。

  ```
  e_set("result", str_lower(v("ret", "return")))
  ```

  d. 示例4：多次设置字段值。

  - 原始数据

    ```
    {
        "ret" : "fail"
      }
    ```

  - 加工规则

    ```
    e_set("event_type", "login event", "event_info", "login host")
    ```

  - 加工结果

    ```
    ret: fail
    event_type: login event
    event_info: login host
    ```

- **更多参考**

  支持和其他函数组合使用。

e_drop_fields

调用e_drop_fields函数删除符合条件的日志字段。

- **函数格式**


```
e_drop_fields(key1, key2, ....,regex=false)
```

- **参数说明**

参数	参数类型	是否必填	说明
key	String	是	日志字段名，可以为正则表达式。当字段名完全满足条件时删除该字段，保留不满足条件的字段。关于正则表达式的更多信息，请参见正则表达式。至少需要配置一个日志字段。
regex	Boolean	否	如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。

- **返回结果**

  返回删除后的日志。

- **函数示例**

  如果content字段的值为123，则删除content字段和age字段。

  – 测试数据

  ```
  {
   "age": 18,
   "content": 123,
   "name": "twiss"
  }
  ```

  – 加工规则

  ```
  e_if(e_search("content==123"), e_drop_fields("content", "age",regex=true))
  ```

  – 加工结果

  ```
  name: twiss
  ```

- **更多参考**

  支持其他它函数组合使用。

e_keep_fields

调用e_keep_fields函数保留符合条件的日志字段。

说明

日志服务中包含内置的元字段，例如__time__、__topic__等。如果在调用e_keep_fields函数时没有保留__time__字段，则日志时间将被重置为系统当前时间。如果您不希望重置元字段的值，需要将元字段放入列表中，常见格式为F_TIME, F_META, F_TAGS, "f1", "f2"。

- **函数格式**

```
e_keep_fields(key1, key2, ....,regex=false)
```

- **参数说明**

参数	参数类型	是否必填	说明
key	String	是	日志字段名，可以为正则表达式。当字段名完全满足条件时保留该字段，删除不满足条件的字段。至少需要配置一个字段。

regex	Boolean	否	如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。

- **返回结果**

  返回保留的日志。

- **函数示例**

  如果content字段的值是123，则保留content和age字段。

  – 测试数据

  ```
  {
   "age": 18,
   "content": 123,
   "name": "twiss"
  }
  ```

  – 加工规则

  ```
  e_if(e_search("content==123"), e_keep_fields("content", "age"))
  ```

  – 加工结果

  ```
  age: 18
  content: 123
  ```

e_pack_fields

用e_pack_fields函数将日志字段进行打包，并输出到新的字段中。

- **函数格式**

  ```
  e_pack_fields(output_fields,include=".*",exclude=None,drop_packed=true)
  ```

- **参数说明**

参数	类型	是否必填	说明
output_field	String	是	指定打包后输出的字段名称。其值为JSON格式。
include	String	否	白名单配置，符合正则表达式的字段会被打包。默认为".*" ，表示全部匹配。
exclude	String	否	黑名单配置，符合正则表达式的字段不会被打包。默认为None，表示不进行匹配判断。
drop_packed	Boolean	否	打包数据后是否删除被打包的原数据，默认为true。 ● true（默认值）：输出结果中删除被打包的原数据。 ● false：输出结果中不删除被打包的原数据。

- **返回结果**

  返回被打包后的日志数据。

- **函数示例**

a. 示例1：将日志所有字段打包到test字段，默认删除被打包的原始字段。

- 测试数据

```
{
 "test1":123,
 "test2":456,
 "test3":789
}
```

- 加工规则

```
e_pack_fields("test")
```

- 加工结果

```
test:{"test1": "123", "test2": "456", "test3": "789"}
```

b. 示例2：将日志所有字段打包到test字段，不删除被打包的原始字段。

- 测试数据

```
{
 "test1":123,
 "test2":456,
 "test3":789
}
```

- 加工规则

```
e_pack_fields("test",drop_packed=false)
```

- 加工结果

```
test:{"test1": "123", "test2": "456", "test3": "789"}
test1:123
test2:456
test3:789
```

c. 示例3：打包test和abcd字段到content字段，不删除被打包的原始字段。

- 测试数据

```
{
 "abcd@#%":123,
 "test":456,
 "abcd":789
}
```

- 加工规则

```
e_pack_fields("content", include="\w+", drop_packed=false)
```

- 加工结果

```
abcd:789
abcd@#%:123
content:{"test": "456", "abcd": "789"}
test:456
```

d. 示例4：不打包test和abcd字段，其余字段打包到content字段，删除被打包的原始字段。

- 测试数据

```
{
 "abcd@#%":123,
 "test":456,
 "abcd":789
}
```

- 加工规则

```
e_pack_fields("content", exclude="\w+", drop_packed=true)
```

- 加工结果


```
abcd:789
content:{"abcd@#%": "123"}
test:456
```

e_rename

调用e_rename函数重命名符合条件的日志字段名称。

- **函数格式**

```
e_rename("key1", "new key1", "key2", "new key2", ..., regex=false)
```

说明

函数中key和new key必须成对出现。

- **参数说明**

参数	参数类型	是否必填	说明
key	String	是	日志字段名，可以为正则表达式。当字段名完全满足条件时，重命名该字段。至少需要配置一个字段。
new key	String	是	重命名后的字段名。
regex	Boolean	否	如果设置为false，表示不使用正则表达式进行匹配。当不配置该参数时，系统默认取值为true。

- **返回结果**

返回重命名后的字段。

- **函数示例**

a. 示例1：将字段host重命名为client_hos。

- 测试数据

```
{
 "host": 1006
}
```

- 加工规则

```
e_rename("host","client_host")
```

- 加工结果

```
client_host: 1006
```

b. 示例2：不存在字段时，不进行重命名。

- 测试数据

```
{
 "host": 1006
}
```

- 加工规则

```
e_rename("url","rename_url")
```

- 加工结果

```
host: 1006
```


- **更多参考**

  支持和其他函数组合使用。

10.2.3.4 事件操作函数

本文介绍事件操作函数的语法规则，包括参数解释、函数示例等。

函数列表

类型	函数	说明
事件操作	e_drop	根据条件判断是否丢弃日志。支持和其他函数组合使用。

	e_keep	根据条件判断是否保留日志。
		e_keep函数和e_drop函数都会丢弃日志。e_keep函数在不满足条件时丢弃，而e_drop函数则是在满足条件时丢弃。
		`# 以下4个加工规则等价`
		`e_if_else(e_search("f1==v1"), e_keep(), e_drop())`
		`e_if_else(e_search("not f1==v1"), e_drop())`
		`e_keep(e_search("f1==v1"))`
		`e_drop(e_search("not f1==v1"))`
		`# 以下加工规则无意义`
		`e_if(e_search("..."), e_keep())`
		`e_keep()`
		支持和其他函数组合使用。

事件分裂	e_split	基于日志字段的值分裂出多条日志，并且支持通过JMES提取字段后再进行分裂。支持和其他函数组合使用。
输出事件	e_output、e_coutput	输出日志到指定的Logstream中，并可配置输出时的topic、source、tag和shard hash信息。 • e_output：执行到e_output函数时，输出日志到指定的Logstream中，且对应的日志不再执行后面的加工规则。 • e_coutput：执行到e_coutput函数时，输出日志到指定的Logstream中，且对应的日志继续执行后面的加工规则。 支持和其他函数组合使用。

e_drop

根据条件判断是否丢弃日志。

- **函数格式**

  ```
  e_drop(condition=true)
  ```

  支持固定标识DROP，等价于e_drop()。

- **参数说明**

参数名称	参数类型	是否必填	说明
condition	Bool	否	默认为true，一般传递一个条件判断函数的结果。

- **返回结果**

  满足条件则丢弃日志并返回None，否则返回原日志。

- **函数示例**

  a. 示例1：当__programe__字段的值为access时丢弃日志，否则保留该日志。

  - 测试数据

    ```
    [
    {
     "__programe__": "access",
     "age":  18,
     "content":  123,
     "name":  "maki"
    },
    {
     "__programe__": "error",
     "age":  18,
     "content":  123,
     "name":  "maki"
    }
    ]
    ```

  - 加工规则

    ```
    e_if(e_search("__programe__==access"), DROP)
    ```

  - 加工结果

    丢弃__programe__字段值为access的日志，保留__programe__字段的值为error的日志。

    ```
    __programe__: error
    age:  18
    content:  123
    name:  maki
    ```

  b. 示例2：条件判断结果为true，丢弃日志。

  - 测试数据

    ```
    {
     "k1":"v1" ,
     "k2": "v2",
     "k3": "k1"
    }
    ```

  - 加工规则

    ```
    e_drop(e_search("k1==v1"))
    ```

  - 加工结果

    因为k1==v1条件为true，因此丢弃该日志。

  c. 示例3：条件判断结果为false，保留日志。

  - 测试数据

    ```
    {
     "k1":"v1" ,
     "k2": "v2",
    ```



```
 "k3": "k1"
}
```

- 加工规则
  ```
  e_drop(e_search("not k1==v1"))
  ```
- 加工结果
  ```
  k1: v1
  k2: v2
  k3: k1
  ```

d. 示例4：不设置判断条件时，使用默认值true，丢弃日志。

- 测试数据
  ```
  {
   "k1":"v1" ,
   "k2": "v2",
   "k3": "k1"
  }
  ```
- 加工规则
  ```
  e_drop()
  ```
- 加工结果

  丢弃日志。

- **更多参考**

  支持和其他函数组合使用。

e_keep

据条件判断是否保留日志。

- **函数格式**
  ```
  e_keep(condition=true)
  ```
  支持固定标识KEEP，等价于e_keep()。
- **参数说明**

参数名称	参数类型	是否必填	说明
condition	Bool	否	默认为true，一般传递一个条件判断函数的结果。

- **返回结果**

  满足条件则返回原日志，不满足时丢弃日志。
- **函数示例**

  a. 示例1：当__programe__字段的值是access的时候保留日志，否则丢弃日志。

  - 测试数据
    ```
    [
    {
    "__programe__": "access" ,
    "age":  18,
    "content":  123,
    "name": "maki"
    },
    {
    "__programe__": "error" ,
    ```



```
"age":  18,
"content":  123,
"name":  "maki"
}
]
```

- 加工规则

```
e_keep(e_search("__programe__==access"))
#等价于
e_if(e_search("not __programe__==access"), e_drop())
#等价于
e_if_else(e_search("__programe__==access"), e_keep(), e_drop())
```

- 加工结果

  保留__programe__字段值为access的日志。

```
__programe__: access
age:  18
content:  123
name:  maki
```

b. 示例2：条件判断结果为true，保留日志。

- 测试数据

```
{
 "k1":"v1" ,
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(e_search("k1==v1"))
```

- 加工结果

```
k1: v1
k2: v2
k3: k1
```

c. 示例3：条件判断结果为false，丢弃日志。

- 测试数据

```
{
 "k1":"v1" ,
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(e_search("not k1==v1"))
```

- 加工结果：丢弃日志。

d. 示例4：判断条件为false。

- 测试数据

```
{
 "k1":"v1" ,
 "k2": "v2",
 "k3": "k1"
}
```

- 加工规则

```
e_keep(false)
```

- 加工结果：丢弃日志。

- **更多参考**

  支持和其他函数组合使用。

e_split

基于日志字段的值分裂出多条日志，并且支持通过JMES提取字段后再进行分裂。

- **函数格式**

  ```
  e_split(字段名, sep=',', quote='"', lstrip=true, jmes=None, output=None)
  ```

  分裂规则：

  a. 如果配置了jmes参数，则将日志字段的值转化为JSON列表，并使用JMES提取值作为下一步的值。如果没有配置jmes参数，则将字段的值直接作为下一步的值。

  b. 如果上一步的值是一个列表或JSON列表格式的字符串，则按照此列表分裂并结束处理。否则使用sep、quote或lstrip将上一步的值进行CSV解析，根据解析后的多个值进行分裂并结束处理。

- **参数说明**

参数名称	参数类型	是否必填	说明
字段名	String	是	需要分裂的字段名。
sep	String	否	用于分隔多个值的分隔符。
quote	String	否	用于引用多个值的配对类字符的引用符。
lstrip	String	否	是否将值左边的空格去掉，默认为true。
jmes	String	否	将字段值转化为JSON对象，并使用JMES提取特定值，再进行分裂操作。
output	String	否	设置一个新的字段名，默认覆盖旧字段名。

- **返回结果**

  返回日志列表，列表中字段的值都是源列表中的值。

- **函数示例**

  – 测试数据

  ```
  {
  "__topic__":,
  "age":  18,
  "content":  123,
  "name":  "maki" ,
  "__topic__":,
  "age":  18,
  "content":  123,
  "name":  "maki"
  }
  ```

  – 加工规则

  ```
  e_set("__topic__", "V_SENT,V_RECV,A_SENT,A_RECV")
  e_split("__topic__")
  ```

  – 加工结果

  ```
  __topic__:  V_SENT
  name:  maki
  age:  18
  ```

```
content: 123
__topic__: V_RECV
name: maki
age: 18
content: 123
__topic__: A_SENT
name: maki
age: 18
content: 123
__topic__: A_RECV
name: maki
age: 18
content: 123
```

- **更多参考**

  支持和其他函数组合使用。

e_output、e_coutput

输出日志到指定的Logstream中，并可配置输出时的topic、source、tag等信息。

- **函数格式**

```
e_output(logsteam, tags=None)
e_coutput(logstream, tags=None)
```

  预览时不会输出日志到目标Logstream中，而是输出到页面，供您调试。

- **参数说明**

参数名称	参数类型	是否必填	说明
Logstream	String	否	输出日志到已存在的Logstream。
tags	Dict	否	为日志设置新的标签，以字典格式传入。

  加工结果

  - e_output：输出日志到指定的Logstream中，且对应的日志不再执行后面的加工规则。
  - e_coutput：输出日志到指定的Logstream中，且对应的日志继续执行后面的加工规则。

- **函数示例**

  说明：因为函数将结果输出到指定日志流，因此无法在执行预览中测试，需要到实际的加工任务中测试。

  a. 示例1：将k2满足正则表达式，输出到target2中。

     - 测试数据

```
{
"__topic__":,
"k1": "v1", ,
"k2":"v2",
"x1":"v3" ,
"x5": "v4"
}
```


- 加工规则

  此处e_drop()函数的作用是把e_if()函数过滤掉的数据做删除处理。如果不添加该函数，则被过滤的数据被投递到默认的存储目标中。

  ```
  e_if(e_match("k2", r"\w+"), e_output("target2"))
  e_drop()
  ```

- 加工结果

  ```
  需要将日志输出到指定日志流，不支持预览
  ```

b. 示例2：将k2满足正则表达式，输出到target2中，并topic设置为topic1。

- 测试数据

  ```
  {
  "__topic__":,
  "k1": "v1", ,
  "k2":"v2",
  "x1":"v3" ,
  "x5": "v4"
  }
  ```

- 加工规则

  ```
  e_if(e_match("k2", r"\w+"), e_output("target2", tags={"topic": "topic1"}))
  e_drop()
  ```

- 加工结果

  ```
  需要将日志输出到指定日志流，不支持预览
  ```

- **更多参考**

  支持和其他函数组合使用。

10.2.3.5 流程控制函数

本文主要介绍流程控制函数的语法规则，包括参数解释、函数示例等。

函数列表

函数	说明
e_compose	用于组合一系列操作。 • 常用于e_if、e_switch、e_if_else中组合操作。 • 依次调用操作，将日志传递转换并返回最后的日志。 • 对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。 支持和其他函数组合使用。

函数	说明

e_if	条件与操作组合。
	• 满足条件则进行对应操作，不满足条件则不进行对应操作，直接进行下一个条件判断。
	• 对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。
	`e_if(`
	`e_has("a"), e_output("target-a"),`
	`e_has("b"), e_output("target-b"),`
	`)`
	例如，该加工规则相当于以下Python代码结构：
	`if e_has("a"):`
	`e_output("target-a")`
	`if e_has("b"):`
	`e_output("target-b")`
	支持和其他函数组合使用。

e_if_else	根据条件判断的结果进行对应操作。
	`e_if_else(e_has("a"), e_output("target-a"), e_output("target-b"))`
	例如，该加工规则相当于以下Python代码结构：
	`if e_has("a"):`
	`e_output("target-a")`
	`else:`
	`e_output("target-b")`

e_switch	条件与操作的组合。
	• 满足条件则进行对应操作并返回结果，不满足条件则不进行对应操作，直接进行下一个条件判断。
	• 如果没有满足任何条件，但配置了默认参数，则执行默认配置的操作并返回结果。
	• 对于某一条日志，如果其中某一操作删除了日志，则不会再执行后续操作。
	`e_switch(`
	`e_has("a"), e_output("target-a"),`
	`e_has("b"), e_output("target-b"),`
	`default=e_output("target-default"),`
	`)`
	例如，该加工规则相当于以下Python代码结构：
	`if e_has("a"):`
	`e_output("target-a")`
	`elif e_has("b"):`
	`e_output("target-b")`
	`else:`
	`e_output("target-default")`
	支持和其他函数组合使用。

e_compose

组合多个操作。

- **函数格式**

  e_compose(操作1, 操作2, ……)

- **参数说明**

参数名称	参数类型	是否必填	说明
操作1	全局操作函数	是	全局操作函数或其组合。
操作2	全局操作函数	否	全局操作函数或其组合。

- **返回结果**

  返回操作后日志。

- **函数示例**

  如果content字段的值为123，则先删除age字段和name字段，然后将content字段的值设置为ctx。

  – 测试数据

  ```
  {
   "content": 123,
   "age": 23,
   "name": "twiss"
  }
  ```

  – 加工规则

  ```
  e_if(
      e_search("content==123"),
      e_compose(e_drop_fields("age|name"),
   e_rename("content", "ctx")),
  )
  ```

  – 加工结果

  ```
  ctx: 123
  ```

- **更多参考**

  支持和其他函数组合使用。

e_if

据判断条件执行操作。

- **函数格式**

  ```
  e_if(条件, 操作)
  e_if(条件1, 操作1, 条件2, 操作2, ……)
  ```

  函数中条件和操作必须成对出现。

- **参数说明**

参数名称	参数类型	是否必填	说明
条件	任意	是	表达式或其组合。其结果不是布尔值时，会进行真假判断。
操作	全局操作函数	否	全局操作函数或其组合。

- **返回结果**

  返回加工处理后的日志。

- **函数示例**

a. 示例1：字段值匹配后再进行操作。

result字段值为failed或failure时，设置__topic__字段的值为login_failed_event。

- 测试数据

```
{
 "result": "failed"
}
```

- 加工规则

```
e_if(e_match("result", r"failed|failure"), e_set("__topic__", "login_failed_event"))
```

- 加工结果

```
result: failed
__topic__: login_failed_event
```

b. 示例2：根据字段值判断后再提取数据。

当request_body字段存在且值非空时，调用字段类操作函数JSON将request_body字段展开成多个值。

- 测试数据

```
{
 "request_body": {\"k1": 100, \"k2\": 200}
}
```

- 加工规则

```
e_if(v("request_body"), e_json("request_body"))
```

- 加工结果

```
request_body: {"k1": 100, "k2": 200}
k1: 100
k2: 200
```

c. 示例3：高级判断后再进行操作。

当valid字段的值为小写failed时，丢弃日志。

- 测试数据

```
{
 "valid":"failed"
}
```

- 加工规则

```
e_if(op_eq(str_lower(v("valid")), "failed"),  e_drop())
```

- 加工结果：丢弃日志

d. 示例4：多个条件按顺序操作。

- 测试数据

```
{
 "valid":"failed"
}
```

- 加工规则

```
e_if(True, e_set("__topic__", "default_login"),  e_match("valid", "failed"), e_set("__topic__", "login_failed_event"))
```

- 加工结果

```
valid: failed
__topic__:login_failed_event
```

- **更多参考**

支持和其他函数组合使用。

e_if_else

根据判断条件的结果执行操作。

- **函数格式**
  ```
  e_if_else(条件, 真时操作, 假时操作)
  ```
- **参数说明**

参数名称	参数类型	是否必填	说明
条件	任意	是	表达式或其组合。其结果不是布尔值时，会进行真假判断。
真时操作	全局操作函数	是	全局操作函数或其组合。
假时操作	全局操作函数	是	全局操作函数或其组合。

- **返回结果**

  返回不同条件对应的操作结果。
- **函数示例**

  如果result字段的值为ok或pass，或者status字段的值为200，则保留日志。
  - 测试数据
    ```
    {
    "result":"ok",
    "status": 400
    }
    {
    "result": "Pass",
    "status": 200
    }
    {
    "result": "failure",
    "status": 500
    }
    ```
  - 加工规则
    ```
    e_if_else(
        op_or(e_match("result", r"(?i)ok|pass"), e_search("status== 200")), e_keep(),e_drop()
    )
    ```
  - 加工结果
    ```
    result: ok
    status: 400
    result: Pass
    status: 200
    ```

e_switch

组合多个条件和操作。

- **函数格式**
  ```
  e_switch(条件1, 操作1, ……, default=None)
  ```

  说明 函数中条件和操作必须成对出现。
- **参数说明**

参数名称	参数类型	是否必填	说明
条件	任意	是	表达式或其组合。其结果不是布尔值时，会进行真假判断。
操作	全局操作函数	是	全局操作函数或其组合。
default	全局操作函数	否	默认的全局操作函数或其组合。没有条件满足时执行该操作。

- **返回结果**

  返回加工处理后的日志。

- **函数示例**

  a. 如果content字段的值为123，则将__topic__字段的值设置为Number。如果data字段的值为123，则将__topic__字段的值设置为PRO。

  - 测试数据

    ```
    {
    "__topic__": ,
    "age": 18,
    "content": 123,
    "name":"maki",
    "data": 342
    }
    {
    "__topic__": ,
    "age": 18,
    "content": 23,
    "name": "maki" ,
    "data": 123
    }
    ```

  - 加工规则

    ```
    e_switch(
        e_search("content==123"),
        e_set("__topic__", "Number", mode="overwrite"),
        e_search("data==123"),
        e_set("__topic__", "PRO", mode="overwrite"),
    )
    ```

  - 加工结果

    ```
    __topic__: Number
    age: 18
    content: 123
    name: maki
    data: 342
    __topic__: PRO
    age: 18
    content: 23
    name: maki
    data: 123
    ```

  b. 通过e_switch语法和e_output语法结合，将符合规则的日志投递到不同的Logstream。其中default=e_drop()，表示将不满足规则的日志丢弃，不做投递处理。若不设置default参数，则表示不满足规则日志都会被投递到配置的第一个Logstream中。

说明

output的加工结果不会显示到加工结果框中。

- 测试数据

```
{
"__topic__": "sas-log-dns" ,
"test": "aa" ,
"__topic__": "aegis-log-network",
"test":"ecs" ,
"__topic__": "local-dns" ,
"test":"sls" ,
"__topic__": "aegis-log-login" ,
"test": "sls"
}
```

- 加工规则

```
e_switch(e_match("__topic__","sas-log-dns"),
e_output(name="target1"),
e_match("__topic__","sas-log-process"),
e_output(name="target2"),
e_match("__topic__","local-dns"),
e_output(name="target3"),
e_match("__topic__","aegis-log-network"),
e_output(name="target4"),
e_match("__topic__","aegis-log-login"),
e_output(name="target5"),
default=e_drop())
```

- **更多参考**

  支持和其他函数组合使用。

10.3 使用定时 SQL 进行日志加工

云日志服务提供定时SQL功能，用于定时分析日志内容。定时SQL支持标准的SQL语法，按照调度规则周期性执行日志分析，并将分析结果存储到目标日志流。

说明

目前此功能仅在华北-北京四、华南-广州、华东-上海一局点支持白名单用户提交工单申请使用，详细操作请参考**提交工单**，其他局点暂不支持该功能。

前提条件

- 已成功采集到日志。
- 对源日志内容已完成结构化配置，具体操作请参考**结构化配置**。

限制条件

最多可创建20个定时SQL任务。

创建定时 SQL

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“日志加工>定时SQL”，单击“创建定时SQL” 。

或在左侧导航栏中选择“日志管理 ”，单击日志组或日志流名称，进入日志流详情页面，单击  ，在弹出页面中，选择“定时SQL”，单击“创建定时SQL”。

说明

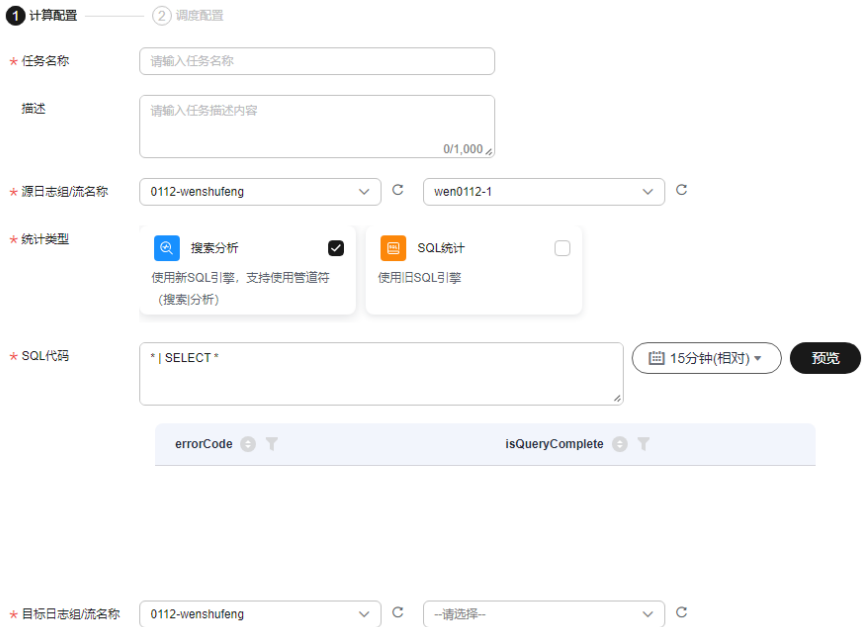
- 通过该方式创建定时SQL时，源日志组/日志流为当前所选的日志组/日志流，且无法修改。
- 通过该方式创建的定时SQL任务，会同步显示在定时SQL列表中，单击任务名称可**查看详细信息**。

步骤3 在创建定时SQL页面中，配置相关参数。

- 在计算配置中，完成如下配置后，然后单击“下一步”。

表 10-8 计算配置参数

参数	说明
任务名称	定时SQL任务的名称。仅支持输入英文、数字、中文、中划线、下划线，且不能以中划线、下划线开头或结尾，长度范围为1~64个字符。
描述	定时SQL任务的描述。长度不超过1000个字符。
源日志组/流名称	选择**已完成结构化配置的日志组/流**，即表示源日志组/流中的日志内容通过定时SQL处理后，将存储到目标日志组/日志流中。选择源日志组/流的时间范围，对该范围内的日志内容进行SQL查询；单击“预览”可查询预览结果。
统计类型	SQL统计（使用旧SQL引擎）
SQL代码	输入的查询和分析语句。 定时SQL运行时，云日志服务将执行该查询和分析语句分析日志。
目标日志组/流名称	存储SQL分析结果的日志组/日志流。



- 在调度配置中，完成如下配置。

表 10-9 调度配置参数

参数	说明
调度间隔	调度定时SQL任务的频率，每调度一次定时SQL任务将产生一个执行实例。调度间隔决定每个执行实例的调度时间。 – 每小时：每隔一小时调度一次定时SQL任务。 – 每天：在每天的某个固定时间点调度一次定时SQL任务。 – 每周：在周几的某个固定时间点调度一次定时SQL任务。 – 固定间隔：按照固定间隔调度定时SQL任务。 – CRON：通过CRON表达式指定时间间隔，按照指定的时间间隔调度定时SQL任务。CRON表达式的最小精度为分钟，格式为24小时制，示例如下： ▪ 0/10 * * * *从00:00开始，每隔整10分钟查询一次，分别为10分钟、20分钟、30分钟、40分钟、50分钟、60分钟。例如：当前时间为16:37，下一次查询时间为16:50。 ▪ 0 0/5 * * *从00:00开始，每隔5小时查询一次，分别为0时、5时、10时、15时、20时。例如：当前时间为16:37，下一次查询时间为20:00。 ▪ 0 14 * * *每天14:00查询一次。 ▪ 0 0 10 * *每月10日00:00查询一次。
调度时间范围	调度的时间范围，具体说明如下： – 某时间开始：实例调度的开始时间，不设时间范围。当该作业被删除时，则不产生新实例。 – 特定时间范围：按照调度间隔生成的实例调度时间必须在该范围内，超出范围则不产生新实例。
起始时间	调度的起始时间。 当调度时间范围选择某时间开始时，需要设置开始时间；当调度时间范围选择特定时间范围时，需要设置起始时间。

参数	说明
SQL时间窗口	定时SQL任务运行时，仅分析该时间范围内的日志。时间窗口表达式的最小精度为秒（s：秒，m：分，h：小时），且SQL时间窗口设置不能超过24小时。可在下拉框中，选择5分钟、15分钟、1小时和1天。如果需要设置其他时间，则通过时间表达式进行设置。 时间表达式为：[+/-{num}h+/-{num}m+/-{num}s@s,+/-{num}@h)。具体说明如下： – “[”表示包含边界。 – “)”表示不包含边界。 – “+”表示时间从当前时间开始往后算起，例如：当前时间是16：00，+1h则表示17：00。这里不使用“+”，对该定时SQL功能无意义。 – “-”表示时间从当前时间往前算起，例如：当前时间时间是16：00，-1h则表示15：00. – “{num}”表示时间取值，为任意整数。逗号前后的时间差值不能超过24小时。 – “@”表示取整。@h：小时取整，分钟和秒忽略不计；@m：分钟取整，秒忽略不计；@s：秒取整。 时间表达式，举例如下： – 5分钟：[-5m@m,@m) – 15分钟：[-15m@m,@m) – 1小时：[-1h@h,@h) – 1天：[-24h@h,@h) – 自定义（不超过一天）：[-65m@h,-5m@h) **说明** 时间表达式最大值为调度间隔的五倍。
SQL超时	执行SQL分析操作失败时，自动重试的阈值。当超时时间超过指定的最大时间和超时次数超过最大次数时，该执行实例结束，状态为失败。 – SQL超时时间取值为60~180秒之间。 – SQL超时次数取值为1~10次之间。

步骤4 完成后，单击“确定”。

----结束

查看定时 SQL

步骤1 在“定时SQL”页签，单击目标作业名称进入详情页面，可查看定时SQL任务的基本信息和执行实例。

说明

每一个实例的上报的日志条数最多为100行。

----结束

修改定时 SQL

步骤1 在“定时SQL”页签，单击目标作业名称操作列对应的“修改”。

或在左侧导航栏中选择“日志管理 ”，选择日志组和日志流。在新版搜索的日志流详情页面，单击右上角  ，在弹出页面中，选择“定时SQL”，单击目标作业名称操作列对应的“修改”。

步骤2 在弹出的页面中，修改**定时SQL参数配置**。

说明

定时SQL的**任务名称**和**源日志组/流名称**无法进行修改。

----结束

删除定时 SQL

步骤1 在“定时SQL”页签，单击目标作业名称操作列对应的“删除”。

或单击目标作业名称，进入定时SQL任务的基本信息页面中，单击右上角“删除”。

步骤2 在弹出的页面中，单击“确定”可删除定时SQL任务。

----结束

10.4 使用 FunctionGraph 服务提供的函数模板进行日志加工

云日志服务提供函数加工，您可以基于函数服务提供的函数模板或者自定义函数，实现日志规整、流转、脱敏、过滤等功能。

说明

目前此功能仅支持华北-北京四、华南-广州、华东-上海一局点，其他局点需要提交工单申请使用。详细操作请参考**提交工单**。

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“日志加工 >函数加工”，进入函数模板页面。

步骤3 根据实际需要，选择函数模板创建函数。

说明

具体的操作步骤，请参见**函数工作流《用户指南》**。

----结束

10.5 日志生成指标（邀测）

支持创建日志指标规则，将上报到LTS的日志数据提取为指标来统一管理，便于后续在应用运维管理控制台的指标浏览、仪表盘界面实时查看数据。

说明

目前此功能在邀测中，支持华北-北京四、华北-乌兰察布一局点，针对白名单用户内测使用，后续将全网开放，敬请期待！

背景信息

- 用户在LTS页面只需按照业务需要创建指标规则即可生成自己的统计报表，设置单个日志过滤条件或通过添加关联关系和添加组设置多个日志过滤条件，保留符合条件的日志，对用户特定时间范围内已结构化的日志进行动态统计，并将统计结果动态呈现到aom的Prometheus实例，操作简单且功能强大。
- 创建的每个指标规则只能生成一个结果，多个结果则需要创建多条指标规则。

前提条件

- 已在应用运维管理控制台创建Prometheus实例。
- 已将日志接入到LTS。
- 已配置结构化数据，当前仅支持已配置结构化的数据进行处理。

说明

日志生成指标要求日志时间的顺序偏差在较小范围内（5s统计频率允许偏差5s，1min统计频率允许偏差1min，5min统计频率允许偏差1min30s），建议优先使用ICAgent结构化方式上报日志，云端结构化方式会引起日志时间乱序严重从而导致无法在统计周期内处理日志，使得统计结果存在偏差。

限制条件

单个用户最多可创建10个日志指标规则，所有规则中添加的指标总数不能超过10。

创建日志指标规则

步骤1 登录云日志服务控制台。

步骤2 在左侧导航栏中选择“日志加工”。

步骤3 在“生成指标”页签，单击“创建规则”。

步骤4 配置日志的基本信息。

1. 填写规则名称，最多256位，只能包含字母、数字、下划线、中划线。
2. 填写描述信息。
3. 启用状态默认打开。
4. 默认勾选委托授权，创建日志生成指标任务，需要您授权LTS和AOM创建云服务委托：lts_admin_trust、aom_admin_trust
5. 任务监控，开启后会将每次任务执行状态写入日志流lts-system/lts-logtometric-statistics，您可以查看日志生成指标任务监控中心或者配置告警规则，及时发现加工过程中可能出现的异常问题。

步骤5 配置数据源。

1. 选择源日志组，若没有，则单击创建日志组。

说明

- 超出存储时间的日志将会被自动删除，您可以按需将日志数据转储至OBS桶中长期存储。
- 如果您的日志尚未接入LTS，请参考**日志接入**，创建日志接入规则，并配置结构化解析规则。

2. 选择日志流，若没有，则单击创建日志流。
3. 日志采样，开启后会对日志源进行随机采样，采样率支持设置为0.1～0.9。

步骤6 配置指标存储位置。

1. 日志生成的指标会被存储到AOM中为自定义指标，请选择要存储的Prometheus实例。若没有，则单击创建实例。
2. 自定义日志生成指标的名称。只支持输入英文、数字、下划线、冒号，且不能以数字、下划线、冒号开头。
3. 填写指标含义。

步骤7 单击“下一步”。

步骤8 在“指标预览”下方预览信息。该预览信息是基于用户配置的日志过滤和统计规则，对日志流执行SQL查询模拟生成的指标结果，依赖用户先将日志采集到LTS，并配置好结构化解析规则和索引配置 ，否则此处预览结果展示为空。

步骤9 配置日志的统计方式。

1. 日志过滤的规则，设置完成后，支持预览效果。如果您无法在日志过滤和日志统计处选择到想要的日志字段，请您先在采集配置中配置好，详细操作请参考**设置云端结构化解析日志**。

   说明

   - 支持“或”“且”两种方式交互式过滤日志。
   - 不同字段支持的过滤规则不同。
   - 日志过滤和日志统计字段的类型仅支持string、float、long，不支持json。
   - 日志过滤是保留符合条件的日志，不符合条件的日志将被丢弃。
   - group by分组字段只支持字符串和整数类型。

2. 设置日志统计的字段，选择统计类型、支持选择或者自定义输入被统计的字段和分组字段。数据迟到1分钟，将不参与统计。

   说明

   - 支持以下统计类型：

     Count: 统计日志条数，CountKeyword: 统计关键词出现的次数， Sum: 统计指定字段求和值，Avg: 统计指定字段平均值，Max: 统计指定字段最大值，Min: 统计指定字段最小值，P50: 统计指定字段50%的值，P75: 统计指定字段75%的值，P90: 统计指定字段90%的值，P95: 统计指定字段95%的值，P99: 统计指定字段99%的值。
   - P系列统计类型是将数字排序后取xx%位置的值作为统计结果。
   - 日志统计是针对符合条件的日志进行操作，执行过程中会在日志组system生成一个日志流，如果删除该日志流将导致所有日志生成指标的规则执行详情无法查看。

3. 选择频率，支持5秒钟、1分钟、5分钟。频率不仅代表上报数据间隔，也代表统计操作的时间窗口，例如：频率5分钟具体含义为到达5分钟间隔后，取当前时间前5分钟数据进行统计操作并上报。
4. 设置完成后，根据查询中选择的内容自动生成维度。
5. 选择单位，例如选择角度、带宽、频率等数据的单位。

步骤10 （可选）单击“实时日志预览”查看实时日志。

步骤11 单击“确定”。

步骤12 创建成功后，在生成指标页签下方，新增一条规则记录。

说明

- 指标可视化和告警：创建成功后，您可以前往应用运维管理AOM控制台的仪表盘配置指标可视化图表，或者前往AOM告警规则配置指标告警。
- 建议每个日志流配置规则数量<= 5个。
- 在对应规则的操作列，支持复制、修改、删除规则。

----结束

11 LTS 配置中心管理

11.1 设置 LTS 日志采集配额和使用量预警

配置超额采集

当日志超过每月免费赠送的额度（500M）时，超过的部分将按需收费。如果每月免费赠送的额度已经可以满足您的使用需求，超过后希望暂停日志收集，可以在配置中心进行设置。

步骤1 在云日志服务管理控制台，单击“配置中心”。

步骤2 选择关闭“超额继续采集日志”。

关闭后表示当日志超过每月免费赠送的额度(500M)时，将暂停采集日志。

说明

- 该开关默认为开启状态，开启后表示当日志超过免费赠送的额度(500M)时，继续采集日志，超过的部分按需收费。
- 云日志服务的计费依据为日志使用量，包括日志读写、日志索引和日志存储。超过免费额度后，如果关闭日志采集开关，将无法再进行日志读写和索引，同时也不再产生日志读写和索引费用。关闭日志采集开关后，超额部分的日志继续存储在LTS，LTS收取日志存储费用，系统将根据配置的日志存储时间老化日志，日志老化后将不再产生任何费用。
- 云日志服务与应用运维管理的日志采集开关为同步状态，即如果您在应用运维管理服务关闭了“超额继续采集日志”开关，则云日志服务的开关也同样关闭。

----结束

配置日志资源使用量预警

开启自定义日志资源使用量预警开关后，系统将自动为您创建一条告警规则（日志资源使用量预警）。当日志使用量超过当前配置的自定义日志资源使用量额度时，系统会发送告警通知。日志使用量包括日志读写流量、索引流量和标准存储量。

步骤1 在“配额设置”页签，单击“自定义日志资源使用量预警”开关。

开启后，云日志服务首页的**资源统计**数据将投递到自动创建的日志流下（lts-system/lts-resource-statistics），可以通过日志告警功能实现自定义日志资源使用量预警。

表 11-1 参数配置说明表

参数	说明
最近1小时日志读写流量	设置最近1小时内云日志服务的日志读写流量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。 当日志读写流量超过设置额度时，会触发日志告警功能。
最近1小时日志索引流量	设置最近1小时内云日志服务的日志索引流量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。 当日志索引流量超过设置额度时，会触发日志告警功能。
日志最新标准存储量	设置最近1小时日志的最新标准存储量额度。默认值为1024GB，单位为GB，只能输入数字和小数点，保留4位小数，且不能以小数点开头和结尾，最小值为0，最大值输入长度不能超过10个字符。 当日志最新标准存储量超过设置额度时，会触发日志告警功能。

说明

- 参数为“或”关系，当任意一个参数满足条件时，告警列表会产生一条告警。
- 该开关默认为关闭状态，当开启后表示当日志资源使用量超过1024GB时，会触发告警规则。
- 开启该开关后，默认会自动创建日志组/日志流（lts-system/lts-resource-statistics）和告警规则（log-statistics-alarm）；关闭该开关，则会自动删除告警规则（log-statistics-alarm）。
- 日志资源使用量会每小时统计一次。
- 系统自动创建一条告警规则（日志资源使用量预警），SQL告警语句为：select write_traffic,index_traffic,storage，条件表达为：write_traffic > 1.099511627776E12 || index_traffic > 1.099511627776E12 || storage > 1.099511627776E13，如果您希望产生的告警以短信、邮件等方式通知您，您可以修改**告警规则**，配置发送通知。
- 如果告警规则不存在，重新开启自定义日志资源使用量预警开关，系统会默认创建规则。

----结束

11.2 设置 LTS 日志内容分词

通过配置分词可将日志内容按照分词符切分为多个单词，在日志搜索时可使用切分后的单词进行搜索。初次使用时，LTS已默认进行了分词配置，默认配置的分词符为：

```
, '";=()[]{}@&<>/:\\?\n\t\r
```

若默认分词符不能满足您的需求时，可按照如下操作进行自定义配置。

注意事项

分词配置只会对配置时间点以后生成的日志生效，之前的日志按照之前配置的分词符进行处理。

配置分词

步骤1 在左侧导航栏中选择“配置中心”，选择“分词配置”页签。

步骤2 配置分词。

LTS提供了如下两种配置分词的方法。若同时使用了这两种配置方法，则分词符取并集。

- 自定义分词符：单击“编辑”，在文本框中输入分词符。
- 特殊分词符：单击“编辑 > 添加特殊分词符”，参考**ASCII码对照表**输入ASCII值。

步骤3 预览分词效果。

在文本框中输入待预览的日志内容，单击“预览”。

步骤4 预览确认配置无误后单击“保存”。

说明

单击“重置”，可恢复到默认分词配置。默认分词符为

```
, '";=()[]{}@&<>/:\\?\n\t\r
```

----结束

ASCII 码对照表

表 11-2 ASCII 码对照表

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
0	NUL（空字符）	32	空格	64	@	96	`
1	SOH（标题开始）	33	!	65	A	97	a

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
2	STX（正文开始）	34	"	66	B	98	b
3	ETX（正文结束）	35	#	67	C	99	c
4	EOT（传输结束）	36	$	68	D	100	d
5	ENQ（询问字符）	37	%	69	E	101	e
6	ACK（确认回应）	38	&	70	F	102	f
7	BEL（响铃）	39	'	71	G	103	g
8	BS（退格）	40	(	72	H	104	h
9	HT（水平定位符号，制表符）	41	)	73	I	105	i
10	LF（换行）	42	*	74	J	106	j
11	VT（垂直定位符号）	43	+	75	K	107	k
12	FF（换页键）	44	,	76	L	108	l
13	CR（归位键）	45	-	77	M	109	m
14	SO（取消变换）	46	.	78	N	110	n
15	SI（启用变换）	47	/	79	O	111	o
16	DLE（跳出数据通讯）	48	0	80	P	112	p
17	DC1（设备控制1）	49	1	81	Q	113	q
18	DC2（设备控制2）	50	2	82	R	114	r
19	DC3（设备控制3）	51	3	83	S	115	s
20	DC4（设备控制4）	52	4	84	T	116	t

ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符	ASCII值	控制字符
21	**NAK（确认失败回应）**	53	**5**	85	**U**	117	**u**
22	**SYN（同步用暂停）**	54	**6**	86	**V**	118	**v**
23	**ETB（区块传输结束）**	55	**7**	87	**W**	119	**w**
24	**CAN（取消）**	56	**8**	88	**X**	120	**x**
25	**EM（连接介质中断）**	57	**9**	89	**Y**	121	**y**
26	**SUB（替换）**	58	**:**	90	**Z**	122	**z**
27	**ESC（跳出）**	59	**;**	91	**[**	123	**{**
28	**FS（文件分割符）**	60	**<**	92	\	124	**|**
29	**GS（组群分隔符）**	61	**=**	93	**]**	125	**}**
30	**RS（记录分隔符）**	62	**>**	94	**^**	126	**~**
31	**US（单元分隔符）**	63	**?**	95	_	127	**DEL（删除）**

11.3 设置 ICAgent 日志采集开关

为了减少内存、数据库和磁盘空间占用，您可以按需进行日志采集设置。日志采集开关用来控制是否对日志数据进行采集。

步骤1 在云日志服务管理控制台，单击“配置中心”，选择“ICAgent采集开关”。

步骤2 单击开启或关闭“ICAgent采集开关”。

说明

采集开关默认打开，当您不需要采集日志时，可通过关闭采集开关来停止日志采集，以减少资源占用。

日志采集关闭后，ICAgent会停止采集日志，且在应用运维管理AOM控制台的“日志采集开关”也会同步关闭。

步骤3 采集Syslog日志到AOM。开关关闭后，ICAgent将不会采集Syslog日志到AOM1.0，此功能仅支持5.12.182以上版本的ICAgent。

步骤4 采集容器标准输出到AOM。选择CCE集群，开启或关闭应用到该集群。开关关闭后，ICAgent将不会采集标准输出日志到AOM，此功能仅支持5.12.133以上版本的

ICAgent。建议使用**云容器引擎CCE应用日志接入LTS**直接采集容器标准输出到LTS，不推荐采集到AOM。

步骤5 ICAgent诊断开关，开关开启后ICAgent运行日志将上报到日志组/日志流lts-system/lts-icagent-statistics，然后您可以使用采集诊断功能查看ICAgent运行状态，及时发现异常情况，此功能仅支持5.12.196及以上版本的ICAgent。详细请参考**采集诊断仪表盘模板**。

----结束

12 查看 LTS 审计事件

概述

云审计服务（Cloud Trace Service，CTS）可以记录LTS相关的操作事件，便于日后的查询、审计和回溯。

开通了云审计服务后，系统开始记录LTS资源的操作。云审计服务管理控制台保存最近7天的操作记录。

开通云审计服务

云审计服务的开通请参见**开通云审计服务**。

开通云审计服务后，如果需要查看LTS相关操作事件，请参见**查询审计事件**。

云审计支持的 LTS 操作列表

云审计支持的LTS操作列表如**表1**所示。

表 12-1 云审计服务支持的 LTS 操作列表

操作名称	资源类型	事件名称
创建日志组	group	createLogGroup
修改日志组	group	updateLogGroup
删除日志组	group	deleteLogGroup
创建日志流	topic	createLogStream
修改日志流	topic	updateLogStream
删除日志流	topic	deleteLogStream
添加日志桶	logPailSetting	addLogPail
删除日志桶	logPailSetting	deleteLogPail
添加日志转储至OBS	als	addLogPailtoObs
修改日志转储至OBS	als	updateLogPailtoObs

操作名称	资源类型	事件名称
删除日志转储至OBS	als	deleteLogPailtoObs
批量启动暂停周期性日志转储	als	batchActionLogPailtoObs
创建指标过滤器	filter	createLogFilter
修改指标过滤器	filter	updateLogFilter
删除指标过滤器	filter	deleteLogFilter
修改指标过滤器状态	filter	updateLogFilterStatus
创建转储	transfer	createLogTransfer
修改转储	transfer	updateLogTransfer
删除转储	transfer	deleteLogTransfer
创建快速查询	searchCriteria	createLogSearchCriteria
修改快速查询	searchCriteria	updateLogSearchCriteria
删除快速查询	searchCriteria	deleteLogSearchCriteria
新增采集路径	logPath	createLogAgent
修改采集路径	logPath	updateLogAgent
删除采集路径	logPath	deleteLogAgent
创建结构化模板	structLogConfig	createLogStreamStructConfig
修改结构化模板	structLogConfig	updateLogStreamStructConfig
删除结构化模板	structLogConfig	deleteLogStreamStructConfig
创建快速分析	wordFreqConfig	updateWordFreqConfig
存储日志路径配置	path	addLogPath
添加统计规则	rule	addRuleStatistics
修改统计规则	rule	updateRuleStatistics
删除统计规则	rule	deleteRuleStatistics
创建结构化	structurization	addStructurization
删除结构化	structurization	deleteStructurization
添加kafka实例	dmsKafka	registerKafkaInfo
修改kafka实例	dmsKafka	updateKafkaInfo
删除kafka实例	dmsKafka	deleteKafkaInfo

操作名称	资源类型	事件名称
创建dis转储	kafkaLogAccess	createDisTransfer
修改dis转储	kafkaLogAccess	updateDisTransfer
删除dis转储	kafkaLogAccess	deleteDisTransfer
创建kafka转储	kafkaTransfer	createKafkaTransfer
修改kafka转储	kafkaTransfer	updateKafkaTransfer
删除kafka转储	kafkaTransfer	deleteKafkaTransfer
创建日志清洗	logFilter	createLogFilter
修改日志清洗	logFilter	updateLogFilter
删除日志清洗	logFilter	deleteLogFilter
创建图表	logChart	createLogChart
修改图表	logChart	updateLogChart
删除图表	logChart	deleteLogChart
创建仪表盘	logDashboard	createLogDashboard
修改仪表盘	logDashboard	updateLogDashboard
删除仪表盘	logDashboard	deleteLogDashboard
打开日志超额采集开关	LogCollectionSwitchOperation	SwitchLogCollectionON
关闭日志超额采集开关	LogCollectionSwitchOperation	SwitchLogCollectionOFF
创建ELB日志桶	elbPailType	createElbPail
修改ELB日志桶	elbPailType	updateElbPail
删除ELB日志桶	elbPailType	deleteElbPail
添加日志路径采集规则	logPathCollectionType	createLogPathCollection
修改日志路径采集规则	logPathCollectionType	updateLogPathCollection
删除日志路径采集规则	logPathCollectionType	deleteLogPathCollection
清理Redis缓存	cleanTenantResourceType	deleteCleanTenantResource

