

目 录

产品简介

核心概念与系统架构

15分钟快速上手

准备工作

第一步：建立你的第一个实体模型

第二步：虚拟数据接入与消息解析

第三步：配置一条超温告警规则

第四步：在低代码设计器中绑定并呈现

操作指南

概念定义

实体管理

产品

网关

请求脚本中使用Auth认证

实体

实体反控

基于HTTP方式进行实体反控

基于WebSocket方式进行实体反控

组合实体

分组管理

组织结构

任务管理

任务

数据推送函数

时序数据查询接口

实体历史属性数据

订阅管理概览

HTTP回调推送

RedisMQ订阅推送

WebSocket订阅推送

系统管理

[导入导出](#)

[日志服务](#)

[低代码设计器](#)

[开发者参考手册](#)

[空间拓扑与ImageMask高级分析](#)

[AI算法的快速应用落地](#)

[附录：BV相机产品Profile介绍](#)

[常见问题与排障](#)

[常见问题与故障排查](#)

[产品性能限流说明](#)

产品简介

产品简介

数字孪生服务 DTService

数字孪生服务 DTService(以下简称DTS)是通过数字孪生模型来模拟和管理现实世界的物理对象、系统或过程的服务，为数字孪生世界提供实时数据采集、分析与预测等功能，实现数字虚拟世界与物理世界的深度融合，帮助组织、企业或机构优化运营、提高效率、减少成本。

BV数字孪生服务可以支持在物理对象对应的实体概念中而不是具体的传感器中查询数据，此服务用于构建可重用、高度可扩展、具有空间感知的体验，以便跨数字世界和物理世界、社交世界将流式处理数据链接到一起。用户可以通过这些独有的相关上下文能力来增强各类上层应用。

DTS功能概览

DTS最核心的两个概念是**实体**和**任务**。实体是数字孪生世界中用户关心的物理或虚拟对象，它将各类数据流组合成为一个具体的事物，也可以是数字原生的虚拟事物，以业务所要求的逻辑进行计算或仿真。产品是一类实体的本体模型。而任务则是自动化规则，在用户给定的触发条件下执行业务逻辑所要求的指令。



实体管理

物理世界与数字模型的建模中介。提供统一的“产品模型-实体对象-网关通道”的本体建模能力。支持高并发的物联网设备或虚拟数据流通过 HTTP/MQTT 等多种协议以及透传认证脚本快速接入，实现设备参数解析、实体状态管理与实体反控的闭环。

BV DTS 数字孪生服务 Digital Twins Service									
实体管理 / 产品									
产品									
<div>创建产品 快捷创建</div> <div>ProductID 请输入ProductID查询 请选择标签</div>									
☐	产品名称	ProductID	描述	所属网关	所属品类	实体数据	创建时间	最近更新	操作
☐	自研无人机	IPVEIIEQy3eGozD	接收地面站无人机编队状态汇总、并...	BV通用网关	自定义品类	4	2026-08-28 18:43:23	2026-08-28 18:43:23	编辑 删除
☐	热电厂热网 配电与环境	g6bFYi6GHQXwJrf	热电厂热网系统中的控制柜和环境监测...	BV通用网关	自定义品类	2	2026-08-28 09:25:07	2026-09-04 10:57:58	编辑 删除
☐	热电厂热网 配电与环境	<AH11mouB7CvA4356vD	热电厂热网系统中的控制柜和环境监测...	BV通用网关	自定义品类	1	2026-08-28 09:25:07	2026-09-04 10:57:58	编辑 删除

产品简介

系统管理	热电厂热泵房 循环水泵	Ed8xb7pej7KcQgpU	热电厂热泵房中的一次/二次循环泵...	BV通用网关	自定义品类	2	2026-08-28 09:25:02	2026-09-04 10:58:19	编辑	删除
	热电厂热泵房 热泵主机	G95T5nfZ6GEv3Jfb	热电厂热泵房中的主机通测与能效告...	BV通用网关	自定义品类	2	2026-08-28 09:24:59	2026-09-04 10:58:28	编辑	删除
	钉钉全公司考勤统计	SBVgIgxAlDekVf2	汇总钉钉考勤组成质的全公司出勤...	BV通用网关	自定义品类	1	2026-08-13 10:20:27	2026-08-13 10:20:27	编辑	删除
	发电机产品	U3IE5qWOT7qaVgd	用于电厂1号发电机组的发电机产品...	BV通用网关	自定义品类	1	2026-08-11 15:14:19	2026-08-11 15:26:34	编辑	删除
	汽轮机产品	HYdbKlonhV5iv3gD	用A电厂1号发电机组的汽轮机产品...	BV通用网关	自定义品类	1	2026-08-11 15:14:17	2026-08-11 15:26:57	编辑	删除
	锅炉产品	OIBY6Gjg0jmgYfbc	用于电厂1号发电机组的锅炉产品...	BV通用网关	自定义品类	1	2026-08-11 15:14:14	2026-08-11 15:27:17	编辑	删除
	ErgoLAB EEG 设备 (LSL 专用)	ZgZi4epscdV8l7m9	接收 ErgoLAB LSL EEG_TimeSeries...	ErgoLAB LSL 接入网关	自定义品类	1	2026-08-01 20:20:42	2026-08-01 20:20:42	编辑	删除

共80条, 每页显示 10 << < 1 > >> GO

收起菜单

任务

孪生系统的规则自动化引擎。支持基于“定时触发、设备属性变更触发、异常事件触发”等多维度的触发机制，结合运行状态、生效时段等安全锁判定条件，自动执行属性修改、服务下发及高级推送脚本逻辑，实现跨物理设备与外部业务系统之间的自动化流转。

BV DTS 数字孪生服务

Digital Twins Service

组织结构

空间拓扑与业务体系的关联中枢。支持以“孪生体”为节点构建空间树状或网状（孪生图）拓扑关系，能够直观映射现实世界中地理空间、组织架构及人机设备之间的复杂从属与位置关联，为精细化空间感知和数据聚合提供结构支撑。

BV DTS 数字孪生服务

Digital Twins Service

just

🏠 首页

🏢 实体管理

📁 分组管理

📁 组织结构

📁 任务管理

组织结构

🔍 搜索孪生体

+

热电厂热泵房 | 1号热泵房

🔗 基本信息

根节点Root / ... / 热电厂热泵房 | 北... / 热电厂热泵房 | 1...

➕ 添加实体

🔍 搜索

🗑️ 删除

🔍 搜索实体

🔍

☐

实体名称

DeviceID

生命周期类型

所属产品

所属网关

加入时间

操作

孪生体虚拟组织

实体数量

📁 界 根节点Root

🔗 系统内置

142

📁 内部 DTS内部结构

🔗 系统内置

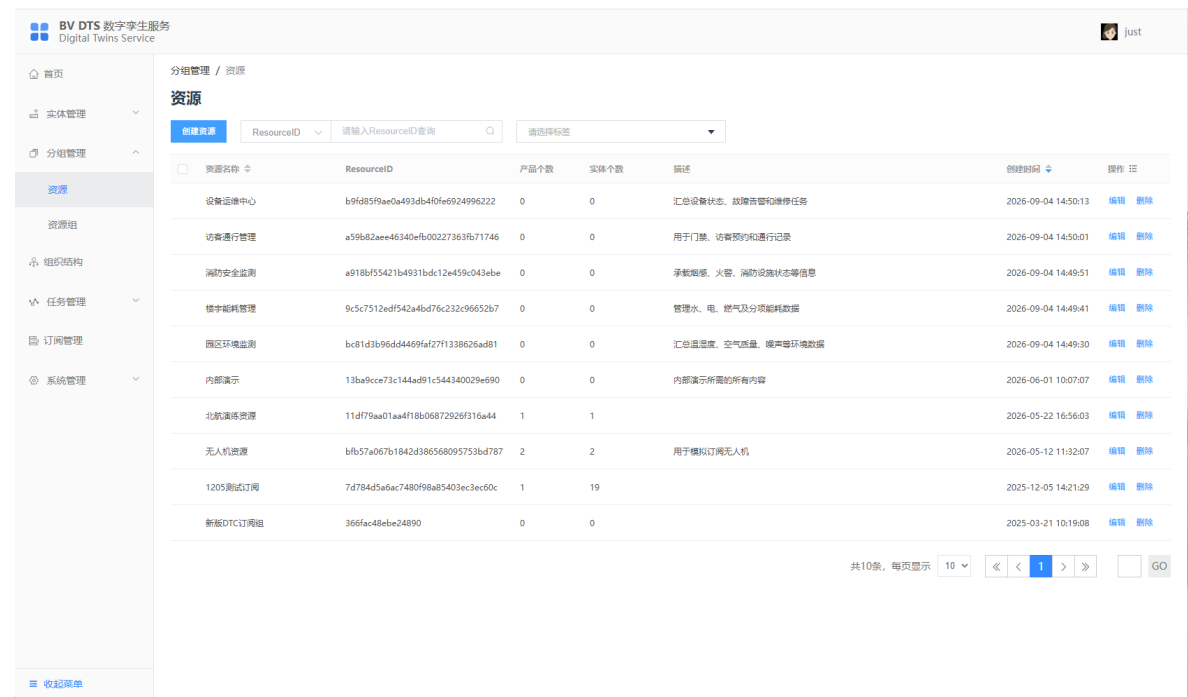
65

产品简介



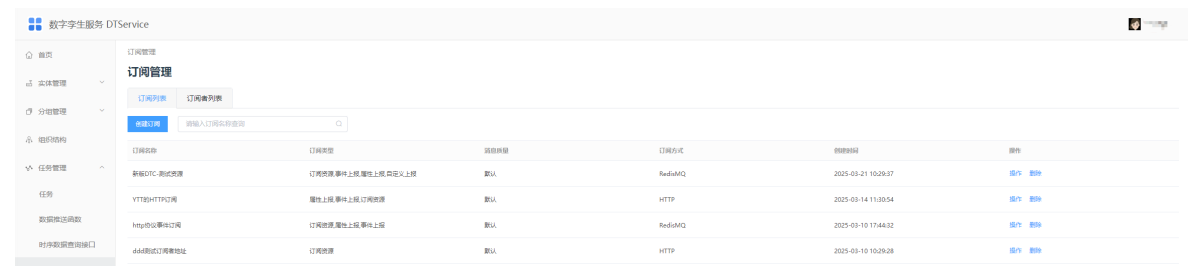
分组管理

跨产品的多维度权限与业务分组。允许根据资产属性、物理位置、使用角色等维度对实体与任务资源进行灵活地贴标分类与资源组归档，可实现多项目团队协作下的跨产品精细化检索与资源访问隔离。



订阅管理

低延迟、高性能的数据流消费分发机制。提供标准的消息订阅服务，允许第三方应用或消费者平台通过高性能 RedisMQ 队列、Websocket或 HTTP 回调（Webhook），即时消费实体上线、状态变更、遥测数据及告警事件，支持高并发环境下的数据流向下游快速流转。



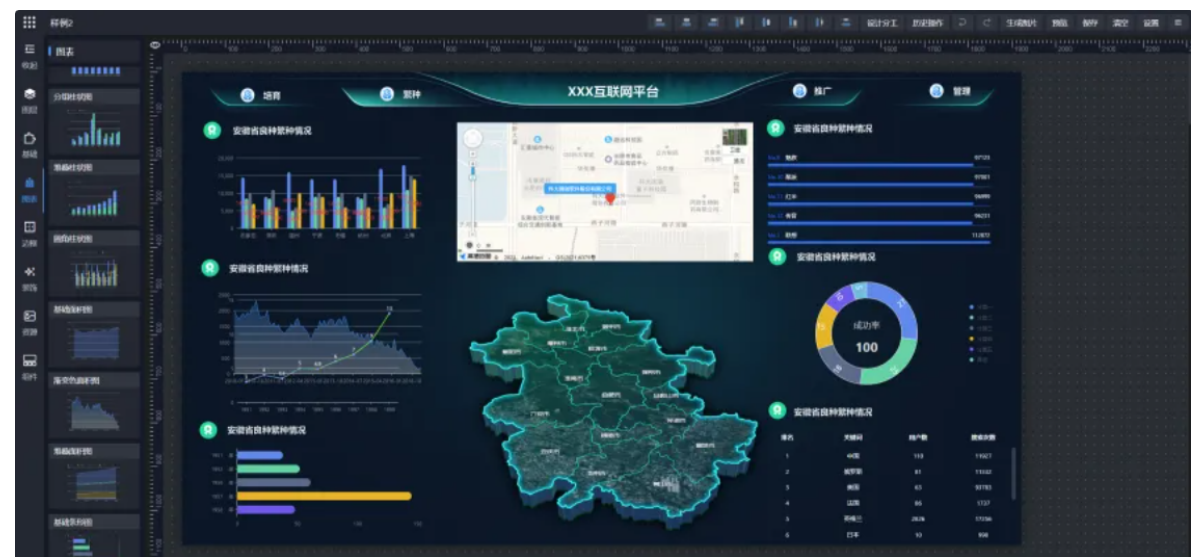
产品简介

订阅管理	VTB订约	订约资源属性上展-事件上展	默认	RedisMQ	2025-03-07 09:46:12	操作 删除
系统管理	订约11	订约资源	默认	RedisMQ	2025-03-06 14:48:53	操作 删除
	测试订约资源	订约资源	默认	HTTP	2025-03-05 15:52:20	操作 删除
	智能决策模型自定义上展Demo	订约资源-自定义上展	默认	RedisMQ	2025-02-14 10:02:57	操作 删除
	智能决策模型属性上展	订约资源-属性上展-事件上展	默认	RedisMQ	2025-02-12 16:46:51	操作 删除
	测试20241218	订约资源-属性上展-事件上展	默认	RedisMQ	2024-12-18 17:08:15	操作 删除

共15条，每页显示 10 < 1 > GO

低代码设计器

孪生数据的一站式低代码可视化看板。作为 DTS 生态中的独立可视化看板编辑器，无缝对接 DTS 产品与实体数据集。提供丰富的 2D/3D 图表组件、布局配置及自由拖拽能力；支持关系型数据库、JSON 接口及 Groovy 脚本等多源数据集绑定，帮助用户零代码快速构建面向数据大屏监视、指挥舱监控等具备实时数据流呈现场景的可视化应用。



附件

- [m_d368685cdc4f1eb5b13fe409908565fa_r.png](#)
- [m_6e9715530fd87c594cfd72f718d8fe9b_r.png](#)
- [m_6db738ee197c59cfc60f269f7b800cd3_r.png](#)
- [m_22c89f47d8bfc6d5085b14100542e948_r.png](#)

核心概念与系统架构

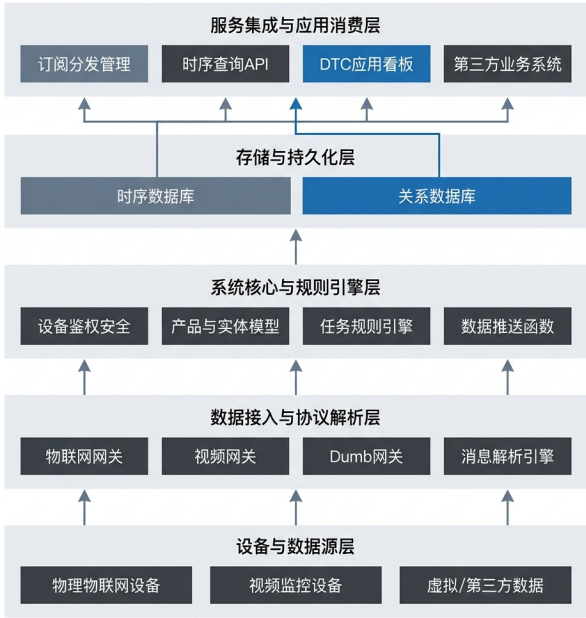
核心概念与系统架构

本节主要介绍 DTS 的核心概念，并展示系统的整体架构设计。

系统架构

DTS（Digital Twin Service，数字孪生服务）采用分层架构设计，实现了从物理/虚拟设备的连接接入、协议解析、孪生建模，到规则引擎计算、数据持久化及上层应用消费的完整闭环。

系统的整体架构图如下所示：



架构各层功能说明

1. 设备与数据源层

- **物理物联网设备：**现场部署的各类传感器、网关采集终端、智能仪表等，负责将物理世界的数据进行采集。
- **视频流媒体设备：**网络监控摄像机、IPC、NVR 等音视频采集设备。
- **虚拟/第三方设备数据：**不具备直接通信能力的虚拟设备、或者是从第三方系统接口拉取的逻辑数据。

2. 数据接入与协议解析层

- **网关通道：**针对不同数据协议提供相匹配的网关服务（物联网网关、视频网关、Dumb网关）。接收设备使用 GatewayKey /

`GatewaySecret` 进行鉴权和安全接入。

- **消息解析**：对于非标准数据格式，通过沙箱运行的消息解析脚本（JavaScript）将上报的透传二进制或非标 JSON 转换为 DTS 系统内标准的 **BV JSON** 数据。

3. 系统核心与规则引擎层

- **孪生本体建模**：通过定义产品 Profile（属性、事件、服务）对实体模型进行本体抽象，并实例化生成具体的数字孪生实体。
- **任务与规则引擎**：配置“触发条件（定时/属性/事件） $\rightarrow$ 生效条件 $\rightarrow$ 执行任务”链条，实现实体状态的自动联动控制和业务流转。
- **数据推送函数**：作为引擎的核心动作执行插件，提供时序数据过滤、Http 请求转发、RedisMQ 订阅通知以及数据归档能力。

4. 存储与持久化层

- **关系数据库**：存放产品 Profile 定义、实体元信息、系统配置信息以及很少变化的“持久型属性”（如固件版本、设备名称）。
- **时序数据库**：高并发写入设备上报的“遥测型属性”历史时序记录（如每秒温度变化）以及事件触发的历史流水，供后续诊断分析使用。

5. 服务集成与应用消费层

- **订阅与分发**：支持将实时消息分发给配置了订阅规则的应用消费者，通道包括 HTTP 回调、RedisMQ 消息队列以及 WebSocket 长连接。
- **查询 API**：向外部业务系统或 DTC 平台提供高性能的历史时序数据查询能力。
- **应用终端**：直接对接 DTC（Digital Twin Center）业务系统及低代码设计器，快速呈现孪生体模型及其实时或历史数据图表。

核心概念

本节主要介绍 DTS 的核心概念。

产品与实体

名词	描述
----	----

名词	描述
产品	一类具有相同功能的实体的集合，产品定义了一类实体的标准属性、事件、服务，其Profile可以理解为该类实体的本体描述。 DTS为每个产品颁发全局唯一的productId。
实体	可以是物理存在的、可通过网络连接的单个物联网设备，也可以是虚拟的设备或其他物体。 DTS为实体颁发、自定义的或从数字孪生服务导入的实体唯一标识deviceId，一般由字母、数字和特殊字符组成，具有配套密钥deviceSecret，由DTS配套生成，一般为8-32个16进制字符。 实体需要通过网关连接DTS。
生命周期	实体生命周期分为 永久实体 和 临时实体 。 永久实体是指长时间存在于系统中的实体，不会被系统主动删除。 临时实体是指短时间存在于系统中的实体，系统通过心跳机制管理其生命周期，存活超时后系统自动删除。

产品描述 Product Profile

产品Profile用于向DTS描述一类实体的共性属性、事件、服务，是该类实体的本体定义。用户通过定义产品Profile，在DTS中构建一类实体的抽象模型并实例化为具体实体，向外提供该实体的数据流服务，使数据消费者容易理解并访问该实体提供的属性、事件、服务功能模型，如颜色、开关等。产品Profile从属性、事件、服务三个方面，分别描述该实体的当前状态、产生事件、对外提供的反控方式。
产品Profile具有预先定义的产品库，在产品库中选择“品牌”及产品可以实现快捷创建。

名词	描述
----	----

名词	描述
产品Profile	<p>对一类实体的功能描述，包括一类实体的属性、服务和事件。</p> <p>通过定义一种物的描述语言来描述产品模型，称之为Profile，采用JSON格式，可以根据产品Profile组装上报实体的数据。</p>
属性	<p>对一个具体状态量、参数量或可配置量的定义，例如温度、速度、位置、模式。</p> <p>例如对于独立的温度和湿度测量时序数据，可以用一个“温湿度传感器”实体的“温度”、“湿度”两个属性分别进行定义，实现实体属性的正向映射。</p> <p>属性有两种更新类型：遥测型属性（默认）：动态或快变数据的更新方式，对数据消费者来说，这些传感器的测量值仅实时有意义，过时的数据就没用了，同时它们的更新频率往往很高，所以默认不存入数据库，仅每个心跳周期内存储第一条数据。对于历史数据流的诊断分析应用，可以选择将其进行时序数据记录，存入专门的时序数据库；持久型属性：静态或慢变属性的更新方式，如设备的固件版本、设备名称，它们一般很少变化，每次属性更新都会存储到数据库。</p>
事件	<p>实体主动发出的、具有时间点意义的事实，例如告警、碰撞、进入区域、任务完成、状态异常。</p> <p>事件是包含事实描述的通知信息，可包含多个输出参数。</p> <p>事件可以被订阅和推送。</p>
服务	<p>外部可对实体发起的操作，实体以服务的方式提供，例如启动设备、切换模式、打开舱门、执行仿真动作。</p> <p>实体的服务是对反向控制接口调用或能力的封装。</p>

网关

网关是 DTS 进行设备数据接入与协议解析的核心通道。网关不能在界面上随意创建；每一个运行的网关在底层都必须有对应的网络协议接收服务支持（例如服务器端运行的 HTTP 或 MQTT 数据接收解析程序），不同的协议（如 HTTP、MQTT、流媒体等）和数据格式需使用相匹配的网关。网关负责聚合与代理其下属子实体的数据流，执行设备鉴权，并处理非标协议解析。

目前，产品需要关联到网关下，才能与 DTS 完成数据交互。DTS 内置提供了一个“通用网关”，支持使用标准 HTTP 协议的实体快速接入。若标准协议无法满足要求，开发者需要根据接入协议定制开发对应的实体接入服务（关于接入协议规范，请参考[实体接入](#)）。

网关根据管理和接入协议类型分为以下三类：

名词	描述
Dumb网关	系统内置的逻辑虚拟网关。 它 不需要任何底层的物理网络通道或协议接收服务 。 专门用于在逻辑上管理无需网络连接的虚拟实体（如组合实体）， 以维持数字孪生模型层次结构的完整性。
物联网网关	依托底层特定的数据通信协议（如 HTTP、MQTT 等）接收服务运行的网关通道。 专门用于代理挂载在该协议下的多个物理子实体， 完成数据的集中上报与指令下发。
视频网关	依托底层的视讯/流媒体处理服务（如系统内置的视频流媒体中间件） 运行的网关通道。专门用于接入和管理流媒体设备（如国标 GB28181、RTSP、WebRTC 等协议的相机、摄像头）， 实现视频流的传输和状态监控。

消息解析

实体映射（数据上行）时，针对采用透传/自定义数据格式的实体，需要在DTS编写消息解析脚本，将实体上报的二进制数据或自定义的JSON数据，转换为BV JSON数据格式；
实体反控（数据下行）时，将DTS下发的BV JSON格式数据，转换为实体或服务支持的格式。

任务

任务是自动化的业务逻辑规则，在满足触发条件时，DTS会自动执行预设的任务，如实现实体事件之间的联动。任务可用于各种复杂场景或具体业务运营场景。针对AI算法，在其服务化的基础上，可以在DTS产

品-服务中进行注册，然后就可以在任务管理中编辑任务，通过调度和使用快速进行落地。

订阅关系

在DTS的所有资源（产品、实体、孪生体等）中与订阅者创建订阅关系，当订阅的资源发生变更的业务信息（如实体数据上报、事件数据上报、属性数据上报、实体状态变化等）时，将所选择资源的消息流转至目标订阅者。

订阅者

订阅者指需要从DTS进行订阅的服务端或应用平台。消费者指在DTS平台为订阅者的应用端创建的扩展体系。在消费者扩展体系内可以创建组织架构、用户组、用户并为用户指定角色。可以根据订阅者服务端或应用平台内部的用户体系创建消费者，配置订阅后，DTS会将资源下所有实体的已订阅类型的消息转发至指定的消费者。

应用流程

在实际业务场景中，使用 DTS 服务通常遵循以下标准流程：

1. **本体建模（定义产品 Profile）**：首先在平台中创建产品，根据实体设备的硬件和业务需求，定义其属性、事件和服务，构建实体的数字孪生本体描述。
2. **网关关联与脚本开发**：将产品关联到对应的接入网关下。如果产品数据格式为透传/自定义，则需编写消息解析脚本（上行映射与下行控制），使平台能与设备端进行数据交互。
3. **实体注册与数据上报**：在产品下单个或批量注册设备实体，获取设备证书（`productId`、`deviceId`、`deviceSecret`）。设备使用证书进行身份鉴权，并向网关发送上报数据请求。
4. **配置自动化任务**：在任务管理中，通过规则引擎配置“触发条件 $\rightarrow$ 生效条件 $\rightarrow$ 执行任务”链条，实现实体状态联动（例如：温度过高触发反控降温）、数据推送转发或时序记录。
5. **数据订阅与可视化对接**：通过订阅管理将实时数据流转发至第三方系统，或在低代码设计器中直接绑定实体数据集，快速搭建和预览数字孪生看板。

附件

- [dts_clean_architecture.png](#)
- [dts_clean_architecture.png](#)

15分钟快速上手

快速上手

准备工作

准备工作

在正式开始数字孪生服务（DTS）开发之前，您需要进行以下准备：

1. 注册并登录账户

访问大视景DTS系统，注册您的账号。暂未开放注册时，需要联系售前人员获得体验账号。

2. 准备数据发送模拟工具

由于您可能没有现成的物理传感器，我们将使用软件模拟的方式产生数据： - **推荐工具**：APIFox (推荐) 或 Postman。 - 您也可以使用编程语言（如 Python、Node.js）发送标准的 HTTP POST 请求。

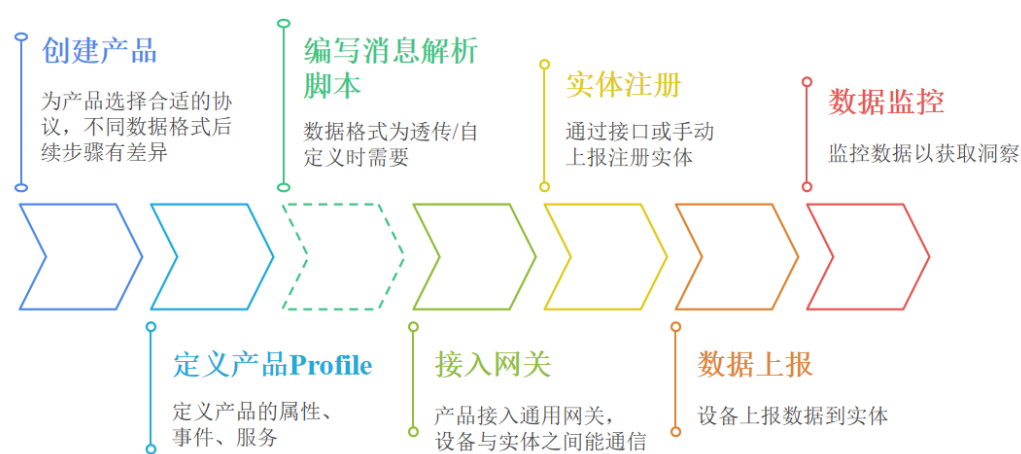
3. 基本术语自检

进入后续步骤前，请确保理解以下核心逻辑关系： - **产品 (Product)**：实体的属性/事件/服务蓝图（本体模型）。 - **实体 (Entity)**：产品的实例化（例如，某特定编号的“温湿度传感器”设备）。 - **网关 (Gateway)**：设备接入的中转接口，负责将设备自定义数据解析成 DTS 能处理的格式。

第一步：建立你的第一个实体模型

实体接入是将虚拟或物理实体无缝集成至DTS，实现对实体的管控。无论是物联网设备（如传感器、智能终端）还是虚拟实体（如软件模拟器），均可通过HTTP协议或定制快速接入，完成数据通信、状态监测及远程控制。实体接入是跨数字世界和物理世界将流式处理数据链接到一起，为构建数字孪生世界提供实时数据来源。

实体接入流程



**准备事项

- 1. 注册数字孪生服务DTService（以下简称DTS）账户
- 2. 准备APIfox，用于模拟设备（或其他可以发送Http请求的软件，如postman、vs等）
- 3. 了解 [BV JSON数据格式](#)
- 3. 了解[HTTP协议接入：BV JSON数据格式](#)、[HTTP协议接入：透传/自定义数据格式](#)

示例

创建产品

- 1. 进入实体管理模块，选择产品。
- 2. 创建一个名为**温湿度传感器**的产品，选择所属品类为“自定义品类”接入网关协议选择为**HTTP协议**，数据格式为**BV JSON**。
 - **自定义品类**：选择自定义产品，本示例将完全自定义产品 Profile。
 - **接入网关协议**：选择通过何种协议方式将数据传输到 DTS 中（本示例使用 **HTTP协议**）。
 - **数据格式**：选择传输到 DTS 中的数据格式（本示例使用平台标准的 **BV JSON**）。
- 3. 根据需要设定产品的坐标系和产品描述；
- 4. 点击“立即创建”，完成产品的创建。

[!NOTE]

第一步：建立你的第一个实体模型

关于品类（标准/自定义）、接入协议（HTTP/定制）以及数据格式（BV JSON/透传）的详细概念定义，请参考[概念定义 - 实体管理](#)。

数字孪生服务 DTService

首页

实体管理

产品

网关

实体

组合实体

低代码设计器

分组管理

组织结构

任务管理

订阅管理

系统管理

实体管理 / 产品 / 创建产品

← 创建产品

* 产品名称

温湿度传感器

* 所属品类

自定义品类

标准品类

* 接入网关协议

HTTP协议

* 数据格式

BV JSON

* 认证方式

实体密钥

位置坐标系

请选择位置坐标系

产品描述

示例产品，采用自定义数据格式17/100

立即创建

取消

定义产品Profile

1. 在产品列表中，选择“温湿度传感器”产品，点击编辑进入产品信息详情页面->选择产品Profile Tab页面。

2. 点击编辑草稿，根据温湿度传感器能提供的属性、事件、服务信息进行产品Profile的定义。

3. 完成后点击退出编辑。

4. 在产品Profile所有编辑完成后，需点击**发布**。发布后的产品Profile才能被网关识别。

下载示例[温湿度传感器产品Profile](#)，修改后缀名为“.json”后导入到产品Profile中，可快速完成产品Profile定义。
事件定义时，勾选持久化存储，接收到的事件将会存储到DTS数据库中。不勾选，暂存在服务器内存中会定时清除。
注意：事件没有订阅者时，DTS不会存储缓存。

数字孪生服务 DTService

首页

实体管理

产品

网关

实体

组合实体

低代码设计器

分组管理

组织结构

任务管理

订阅管理

系统管理

实体管理 / 产品 / 温湿度传感器

← 温湿度传感器

ProductID MNGRcyIFmLxvJh 复制ProductSecret ***** 查看实体数 0 前往管理

产品信息

Topic类列表

产品Profile

消息解析

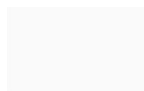
实体详情页

订阅关系

当前展示的是已发布的功能定义，如需修改，请点击 编辑草稿

导出Profile

功能类型	功能名称	标识符	数据类型	数据定义	操作
属性	当前湿度 自定义	humidity	float(单精度浮点型)		删除
属性	当前温度 自定义	temperature	float(单精度浮点型)		删除
属性	电池电量 自定义	batteryLevel	int32(整数型)		删除
服务	设备重置接口 自定义	resetDevice	-		删除
事件	低电量告警 自定义	batteryLowAlarm	-	事件类型:告警	删除
事件	低湿告警 自定义	lowHumidityAlarm	-	事件类型:告警	删除
事件	低温告警 自定义	lowTemperatureAlarm	-	事件类型:告警	删除
事件	高湿告警 自定义	highHumidityAlarm	-	事件类型:告警	删除
事件	高温告警 自定义	highTemperatureAlarm	-	事件类型:告警	删除



编写消息解析脚本（透传/自定义格式）

消息解析脚本是将设备上报数据，转换为BV JSON数据格式；将下发的BV JSON格式数据，转换为设备支持的格式。在实体接入过程中，只有数据格式为**透传/自定义数据**的产品需要编写实体映射（上行）脚本，数据格式为BV JSON 的产品可跳过此步骤。

1. 在产品列表中，选择“温湿度传感器”产品，点击编辑进入产品信息详情页面->选择消息解析 Tab页面。
2. 在实体映射（上行）脚本中，写入将设备产生的数据转化为DTS 可识别的BV JSON数据。
3. 脚本编写完成后，可通过模拟输入检查脚本是否正确，格式转换是否成功，能否正确进行赋值。

调试技巧：使用console.log()将内容输出到日志查看区。运行测试，查看控制台输出；

消息解析脚本示例

```
/**
 * 将透传/自定义数据格式转换为DTS平台的BV JSON数据格式，实体注册、属性、事件上报到DTS时调用。
 * @param {JSONObject} body - 入参
 * @param {JSONObject} body._bv_callback_message - 服务回调参数（回调时携带）
 * @param {string} body._bv_callback_message.productId - 产品编号
 * @param {string} body._bv_callback_message.deviceId - 实体编号
 * @param {string} body._bv_callback_message.identifier - 事件的唯一标识符（回调方式为事件上报时携带）
 * @param {JSONObject} body._bv_callback_message._bv_service_callback - 服务调用参数
 * @param {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * @param {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识符
 * @param {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * @param {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * @param {number} flag - 入参类型标识，0=实体注册，1=属性上报，2=事件上报
 * @returns {JSONObject} 出参，BV JSON数据格式，不能为空
 */
function rawDataToProtocol(body, flag) {
    var inputData = body;
    var jsonObj = {};
    if (flag === 0) {
        /**
         * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
         * @returns {string} return.deviceName - 实体名称（最大长度255），必传
         * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
         * @returns {number} return.deviceType - 实体类型，0=动态资源，1=静态资源，必传
         * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
         * @returns {string} return.version - 版本号（默认default），非必传
         * @returns {JSONObject} return.payload - 扩展数据，用于更新实体属性值，数据基于产品 Profile 定义的属性过滤（已定义属性对应的数据保留，未定义属性对应的数据弃用），非必传
         */
        jsonObj['deviceId'] = inputData.guid;
        jsonObj['deviceName'] = inputData.name;
        jsonObj['deviceStatus'] = inputData.status;
        jsonObj['payload'] = inputData;
        return jsonObj;
    }
    if (flag === 1) {
        /**
         * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
         * @returns {string} return.deviceSecret - 实体签名（不传时将系统匹配），非必传
         */
    }
}
```

第一步：建立你的第一个实体模型

```
* @returns {string} return.deviceName - 实体名称（最大长度255），非必传
* @returns {number} return.deviceStatus - 实体状态， 0=禁用, 1=启用（默认），非必传
* @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
* @returns {number} return.timestamp - 上行数据传输时间戳。格式为数值，值为从UTC 1970-01-01 00:00:00开始到当前时刻的毫秒数，非必传
* @returns {string} return.version - 版本号（默认default），非必传
* @returns {JSONObject} return.payload - 扩展数据，用于更新实体属性值，数据基于产品 Profile 定义的属性过滤（已定义属性对应的数据保留，未定义属性对应的数据弃用），非必传
*/

    jsonObj['deviceId'] = inputData.guid;
    jsonObj['deviceStatus'] = inputData.status;
    jsonObj['payload'] = inputData;
    return jsonObj;
}
if (flag === 2) {
    /**
    * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
    * @returns {string} return.deviceSecret - 实体签名（不传时将系统匹配），非必传
    * @returns {string} return.deviceName - 实体名称（最大长度255），非必传
    * @returns {number} return.deviceStatus - 实体状态， 0=禁用, 1=启用（默认），非必传
    * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
    * @returns {string} return.identifier - 事件的唯一标识符，必传
    * @returns {number} return.timestamp - 上行数据传输时间戳。格式为数值，值为从UTC 1970-01-01 00:00:00开始到当前时刻的毫秒数，非必传
    * @returns {string} return.version - 版本号（默认default），非必传
    * @returns {JSONObject} return.payload - 扩展数据，事件的输出参数，数据基于产品 Profile 定义的事件输出参数过滤（已定义输出参数对应的数据保留，未定义输出参数对应的数据弃用），非必传
    */
    jsonObj['deviceId'] = inputData.guid;
    jsonObj['identifier'] = inputData.notificationType;
    jsonObj['payload'] = inputData.content;
    jsonObj['timestamp'] = inputData.timestamp;
    return jsonObj;
}
}
```

模拟输入示例（设备上报）

```
{
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "name": "温湿度传感器1751939165",
  "status": 1,
  "humidity": 34.54,
  "temperature": 9.38,
  "batteryLevel": 81
}
```

运行结果示例

```
{
  "code": 0,
  "message": "success",
  "info": {
    "productId": "Re8agKekkXw80czi",
    "payload": {
      "temperature": 9.38,
      "humidity": 34.54,
      ...
    }
  }
}
```

第一步：建立你的第一个实体模型

```
        "batteryLevel": 81
    },
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "deviceName": "温湿度传感器1751939165",
    "deviceStatus": 1
  }
}
```

模拟输入示例（属性上报）

```
{
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "status": 1,
  "humidity": 44.97,
  "temperature": 19.05,
  "batteryLevel": 62
}
```

运行结果示例

```
{
  "code": 0,
  "message": "success",
  "info": {
    "payload": {
      "temperature": 19.05,
      "humidity": 44.97,
      "batteryLevel": 62
    },
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "deviceStatus": 1
  }
}
```

模拟输入示例（低温事件上报）

```
{
  "deviceType": "SmartRefrigerator",
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "notificationType": "lowTemperatureAlarm",
  "content": {
    "temperature": -1.41
  },
  "timestamp": 1751943449804
}
```

运行结果示例

```
{
  "code": 0,
  "message": "success",
  "info": {
    "identifier": "lowTemperatureAlarm",
    "payload": {
      "temperature": -1.41
    },
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "timestamp": 1751943449804
  }
}
```

第一步：建立你的第一个实体模型

```
    timesCamp : 1751943449004
  }
}
```

附件

- [温湿度传感器.txt](#)

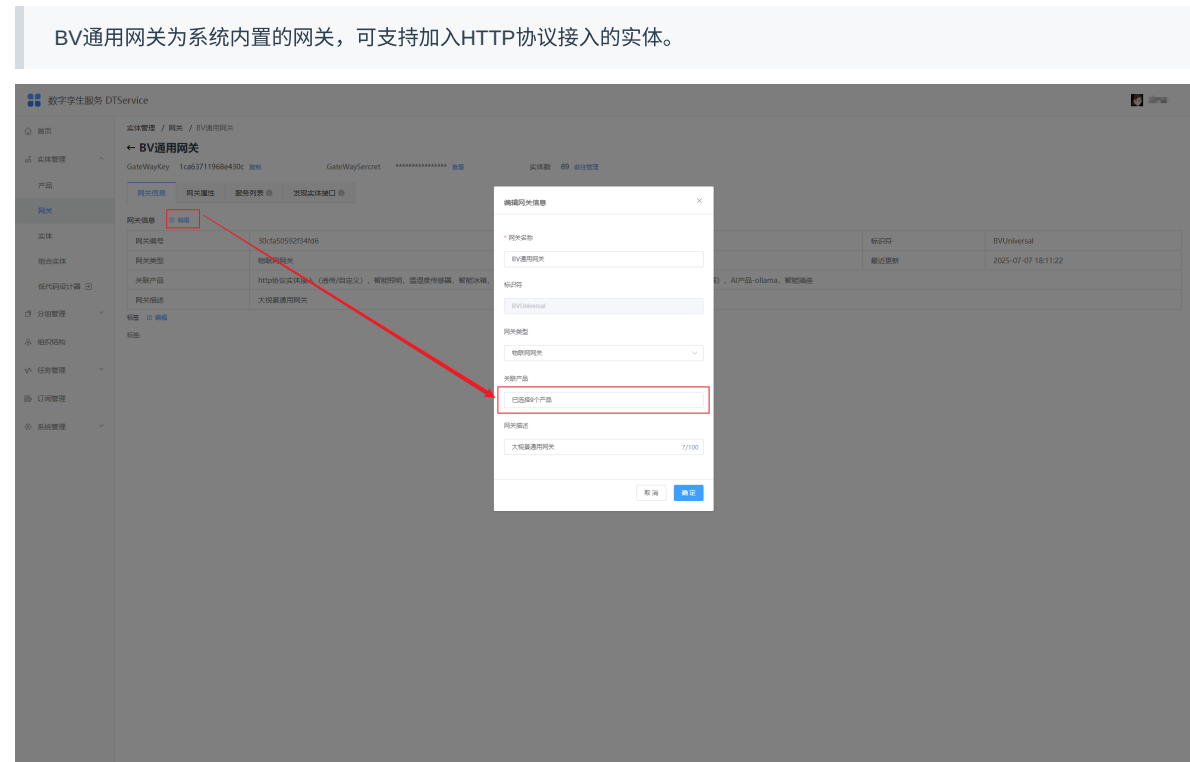
第二步：虚拟数据接入与消息解析

虚拟数据接入与数据上报

DTS 支持多种方式进行数据模拟接入。在此，我们利用 APIFox (或其他 HTTP/MQTT 工具) 来模拟物理传感器，向 DTS 上传测点数据以进行接入测试。

接入网关

1. 进入实体管理模块，选择网关。
2. 选择“BV通用网关”，点击编辑进入BV通用网关的详情页面->选择网关信息 Tab页面。
3. 点击编辑，进入网关信息编辑页面。
4. 修改关联产品，勾选中温湿度传感器产品。将新创建的产品加入到**BV通用网关**下。



实体上报注册

1. 若数据格式为BV JSON，根据[HTTP协议实体接入\(BV JSON数据格式\)-设备上报接口](#)，在APIFox中编写温湿度传感器设备上报数据接口。
2. 接口准备完毕后，向DTS发送实体上报注册的请求。
3. 返回成功，表示实体上报注册完成，便可在DTS 实体页面查找到对应实体。

`productId` 需要从DTS平台产品信息详情中获取。 `deviceId` `deviceName` 为温湿度传感器设备的设备ID和设备名称。

BV JSON发送请求体示例

第二步：虚拟数据接入与消息解析

```
[
  {
    "productId": "MNGRcyIFFmLxxvJh",
    "deviceId": "2d4523e5-cd85-436d-818a-c5629a180349",
    "deviceName": "温湿度传感器1751884069",
    "deviceStatus": 0,
    "payload": {
      "humidity": 0.7952224183,
      "temperature": 0.552,
      "batteryLevel": 85
    }
  }
]
```

若数据格式为透传/自定义，则需根据 [HTTP协议实体接入\(透传/自定义\)-设备上报接口](#)进行实体上报注册。

透传/自定义发送请求体示例

```
[
  {
    "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "name": "温湿度传感器1751939165",
    "status": 1,
    "humidity": 34.54,
    "temperature": 9.38,
    "batteryLevel": 81
  }
]
```

数字孪生服务 DTService

首页

实体管理

产品

网关

实体

组合实体

低代码设计器

分组管理

组织结构

任务管理

订阅管理

系统管理

实体管理 / 实体

实体

产品: 请选择产品 网关: 请选择网关 实体总数 704

添加实体

批量添加

DeviceID

请输入DeviceID查询

请选择标签

☐	实体名称	DeviceID	所属产品	启用状态	创建时间	最近更新	操作
☐	温湿度传感器1751884069	2d4523e5-cd85-436d-818a-c5629a180349	温湿度传感器	<div><div></div>未启用</div>	2025-07-07 18:27:50	2025-07-07 18:27:50	编辑 删除
☐	室外采集器01	1b76EE26D87AB6DD866E6E27D265E712	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	三角支架测试球机	7061396DF993C94960DB83842D39DDF3	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	办公区变像相机	CC30EE7859ESA1F7AC9F705AE4463563	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室内办公区04	229884EB68B15C0C7AB8B6538BF6B78A	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室内办公区01	0545929488F82E3D9E6ECF8DAC15752D	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室内相机07(广角)	68948EFCABCF519C59C12CDD558BA956	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室内相机08(人脸识别)	E9C9939CSA306175831FFC63DD40CA03	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室外采集器05	B467268609CC4EE32FD6FF3369D32C31	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除
☐	室外采集器03	034EB8D06D4A7FB50847E31992F7E380	BV相机-公司环境	<div><div></div>已启用</div>	2025-07-07 17:46:58	2025-07-07 17:46:59	编辑 删除

☐

删除

禁用

启用

共704条, 每页显示 10 < < 1 > > GO

数据上报：属性上报

1. 若数据格式为BV JSON，根据[HTTP协议实体接入\(BV JSON数据格式\)-属性上报接口](#)，在APIFox中编写温湿度传感器设备属性上报数据接口。

2. 接口准备完毕后，向DTS发送属性上报的请求。

3. 返回码200表示成功，表示属性上报成功。其他错误码需根据返回提示进行修改后再次上报。

productId

需要从DTS平台产品信息详情中获取。

第二步：虚拟数据接入与消息解析

BV JSON发送请求体示例

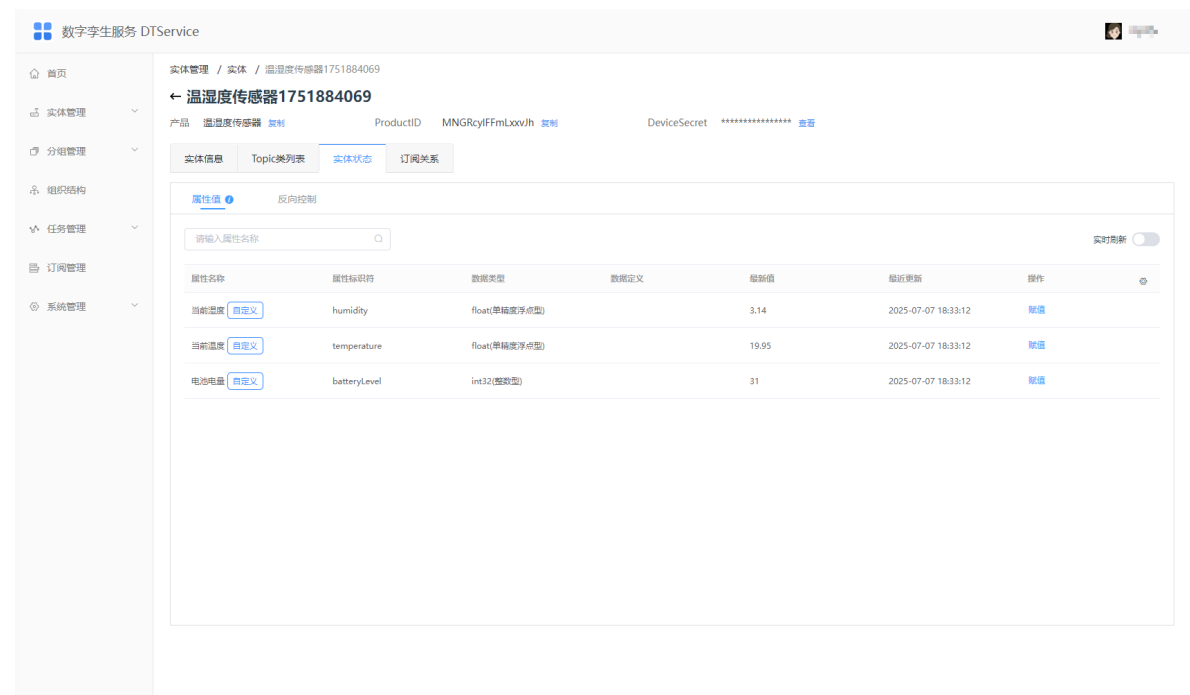
```
{
  "method": 0,
  "productId": "MNGRcyIFFmLxxvJh",
  "deviceId": "2d4523e5-cd85-436d-818a-c5629a180349",
  "deviceStatus": 1,
  "payload": {
    "humidity": 3.14,
    "temperature": 19.95,
    "batteryLevel": 31
  }
}
```

若数据格式为透传/自定义，则需根据[HTTP协议实体接入\(透传/自定义\)-属性上报接口](#)进行属性上报。透传/自定义发送请求体示例

```
{
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "status": 1,
  "humidity": 44.97,
  "temperature": 19.05,
  "batteryLevel": 62
}
```

状态查看

- 1. 进入实体管理模块，选择实体。
- 2. 选择“温湿度传感器1751884069”，点击编辑进入实体详情页—>选择产品profile Tab页；
- 3. 选择运行状态 Tab页，进入属性值的查看。检查设备属性内容是否一致。



数据上报：事件上报

- 1. 若数据格式为BV JSON，根据[HTTP协议实体接入\(BV JSON数据格式\)-事件上报接口](#)，在APIFox中编写温湿度传感器设备事件上报数据接口。
- 2. 接口准备完毕后，向DTS发送事件上报的请求。

`productId` 需要从DTS平台产品信息详情中获取。 `identifier` 为产品Profile定义事件时，设置的事件标识符。

BV JSON发送请求体示例

```
{
  "method": 1,
  "productId": "MNGRcyIFFmLxxvJh",
  "deviceId": "2d4523e5-cd85-436d-818a-c5629a180349",
  "identifier": "lowTemperatureAlarm",
  "deviceStatus": 0,
  "payload": {
    "temperature": -3.56
  }
}
```

若数据格式为透传/自定义，则需根据[HTTP协议实体接入\(透传/自定义\)-事件上报接口](#)。

透传/自定义发送请求体示例

```
{
  "deviceType": "SmartRefrigerator",
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "notificationType": "lowTemperatureAlarm",
  "content": {
    "temperature": -11.95
  },
  "timestamp": 1751941555991
}
```

附件 * [温湿度传感器.txt](#)

第三步：配置一条超温告警规则

实体告警消息实时发送通知到钉钉

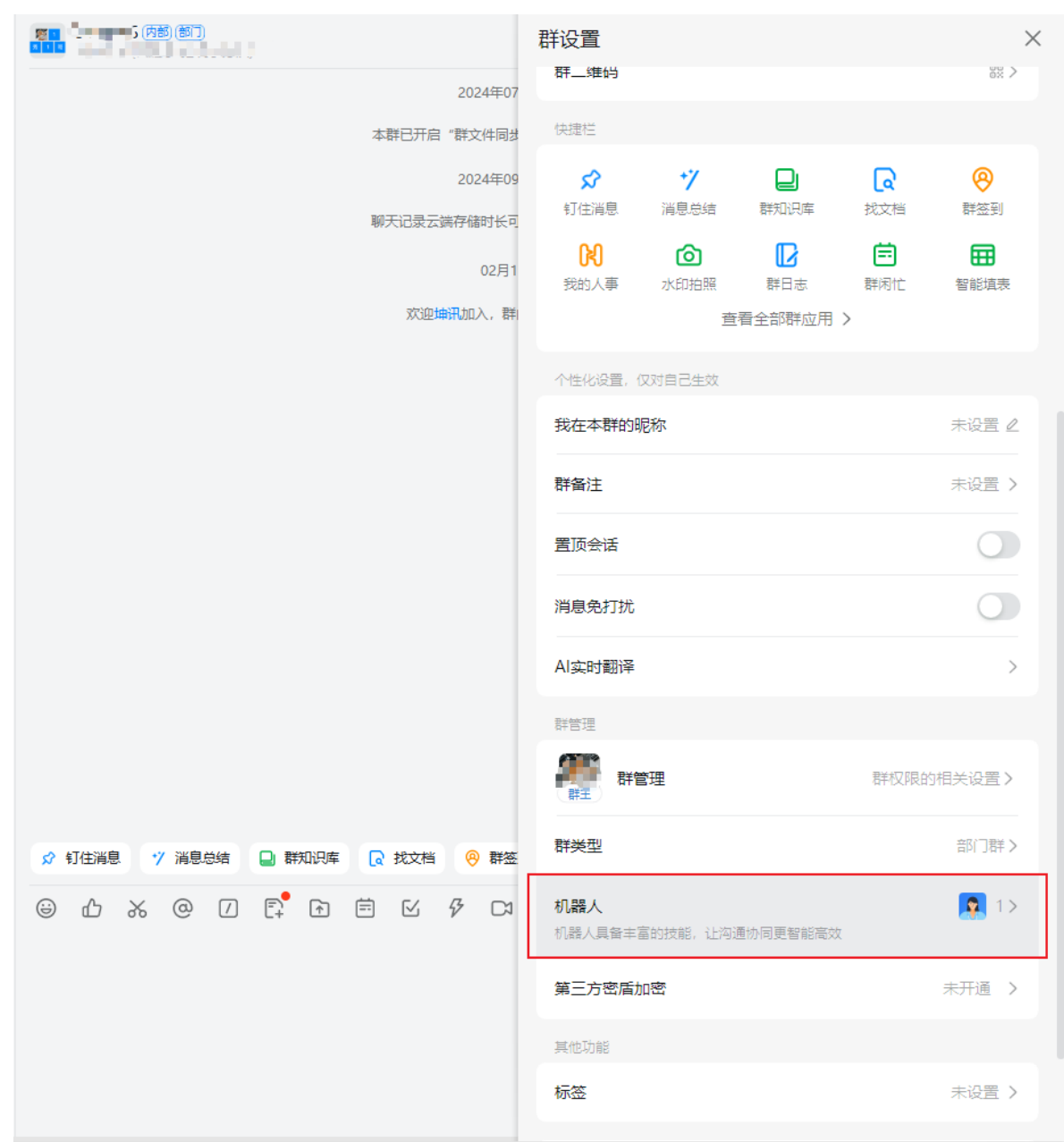
本文将为你介绍如何使用数据推送函数，编写的自定义数据处理的规则与推送逻辑的脚本将实体告警的消息推送到钉钉。

示例

创建钉钉通知机器人

打开钉钉应用，选择需要收取消息的群。进入群设置，选择机器人。

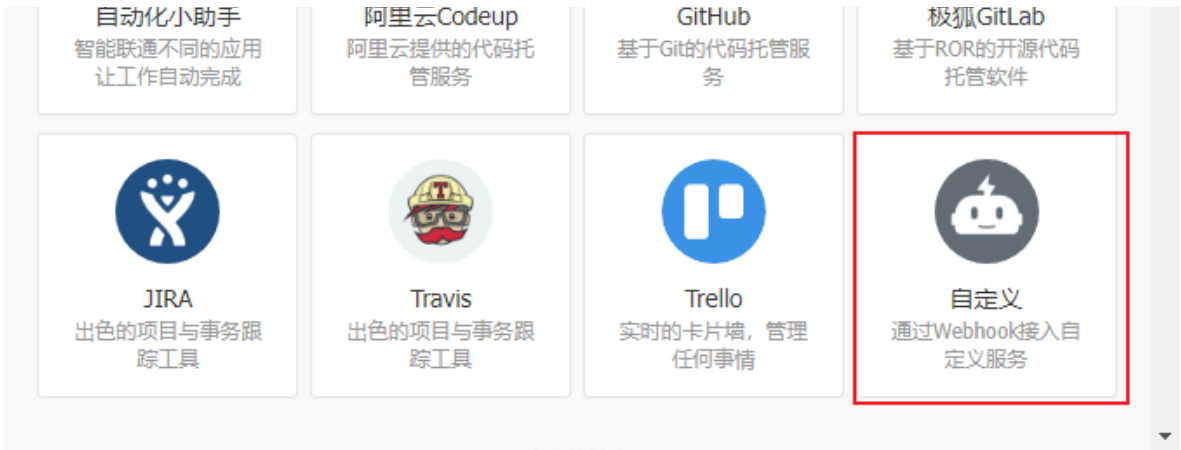
第三步：配置一条超温告警规则



选择添加一个类型为自定义的机器人。



第三步：配置一条超温告警规则



设置机器人的名字，安全设置方式选择自定义关键词的方式，输入设置的关键词。设置完成后，点击保存。(也可以设定为其他方式，对应的请求内容同步修改即可)

添加机器人

机器人名字:

机器人DTS

* 添加到群组:

543

* 安全设置 ?

说明文档

☒ 自定义关键词

告警

第三步：配置一条超温告警规则

消息

添加 (最多添加 10 个)

☐ 我已阅读并同意 [《自定义机器人服务及免责条款》](#)

取消

完成

添加完成，保存好webhook地址已备用。

添加机器人



1.添加机器人✓

2.设置webhook，点击设置说明查看如何配置以使机器人生效

Webhook:

`https://oapi.dingtalk.com/robot/send?access_token`

复制

* 请保管好此 Webhook 地址，不要公布在外部网站上，泄露有安全风险

使用 Webhook 地址，向钉钉群推送消息

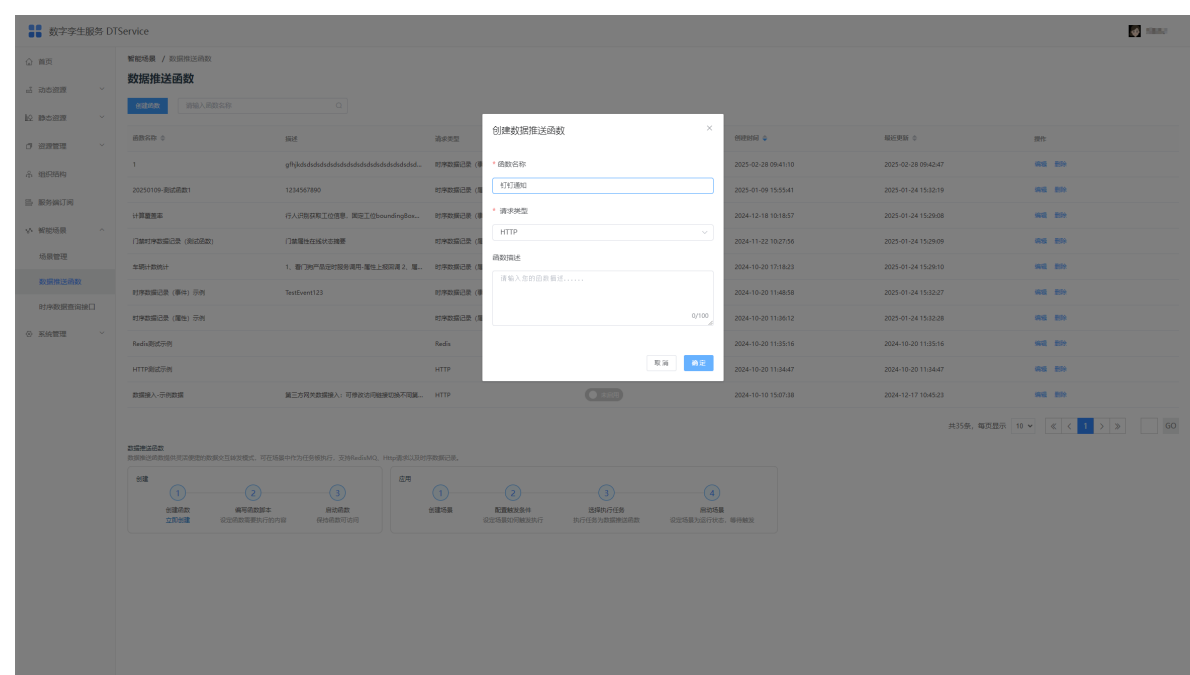
完成

设置说明

创建数据推送函数

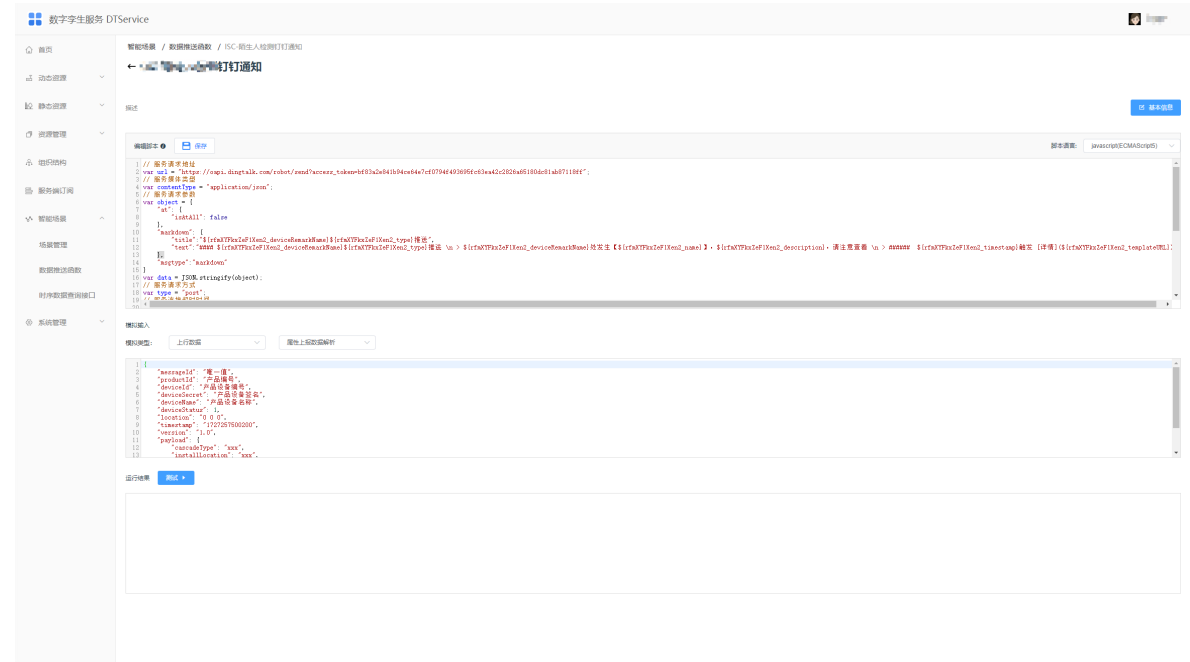
在任务管理-数据推送函数下，创建一个名为“钉钉通知”的函数。创建完成后，进入编辑脚本页面。

第三步：配置一条超温告警规则



编辑函数脚本

根据[钉钉开放文档——自定义机器人发送群消息API接口](#)，编写钉钉通知发送的内容。保存后将函数保持启用状态。



```
/**
 * Http数据交互转发模式
 * 请求脚本已经内置参数：
 * @in-Object {Object} req - 内置函数对象，支持Http同步调用
 * @function {function} req.open(method,url) - 打开一个新的请求
 * @function-param {string} method - HTTP method，必传
 * @function-param {string} url - 请求地址，必传
 * @function {function} req.setRequestHeader(header,value) - 设置请求的 HTTP 头，非必传
 * @function-param {string} header - 请求头key
 * @function-param {string} value - 请求头value
 * @function {function} req.send(data) -发送请求
 * @function-param {string} data -请求体，非必传
 * @response {int} req.readyState -获取请求的状态
 * @response {string} req.responseText - 获取响应文本
 * @response {string} req.status - 返回响应的HTTP状态码
```

第三步：配置一条超温告警规则

```
@response {int} req.status - 获取响应的HTTP状态码
* @response {string} req.statusText - 获取响应的状态文本
* @in-param {JsonObject} body - 内置参数，上行数据的固定属性，可用于计算后二次转发
* @param {string} body.messageId - 消息编号
* @param {string} body.productId - 产品唯一标识
* @param {string} body.productName - 产品名称
* @param {string} body.sourceType - 资源类型 0 动态资源 1 静态资源
* @param {string} body.deviceId - 设备编号
* @param {string} body.deviceSecret - 设备密钥，与deviceId同时存在
* @param {string} body.deviceName - 设备名称
* @param {string} body.deviceStatus - 设备状态 0 未启用 1 已启用
* @param {string} body.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation）
* @param {string} body.identifier - 事件标识符，事件上报时存在，否则Val为空
* @param {string} body.name - 功能名称，事件上报时存在，否则Val为空

* @param {string} body.type - 事件类型 0 信息 1 告警，事件上报时存在，否则Val为空
* @param {string} body.description - 事件描述，事件上报时存在，否则Val为空
* @param {string} body.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数
* @param {string} body.templateId - 消息（事件、属性详情）模板唯一标识符
* @param {string} body.templateURL - 消息（事件、属性详情）模板访问链接
* @param {string} body.version - 版本号。若不传入此参数，则默认default
* @param {JsonObject} body.payload - 验证解析后的上行数据
*/

var alarmType = body.type == 1 ? '告警' : '信息';
var sendMessage = {
  "at": {
    "isAtAll": false
  },
  "markdown": {
    "title": body.deviceName + alarmType + "推送",
    "text": "### " + body.deviceName + alarmType + "推送\n>" + body.deviceName + "处发生【" + body.name + "】，请注意查看\n> ##### " + new Date() + "
  },
  "msgtype": "markdown"
}

req.open("POST", "https://oapi.dingtalk.com/robot/send?access_token=e362881597d97b44d1a502e820875cc0a5c336f4f3f8fa8d7ed91ea0dfc85421", true);
req.setRequestHeader("Content-Type", "application/json");
req.send(JSON.stringify(sendMessage));
```

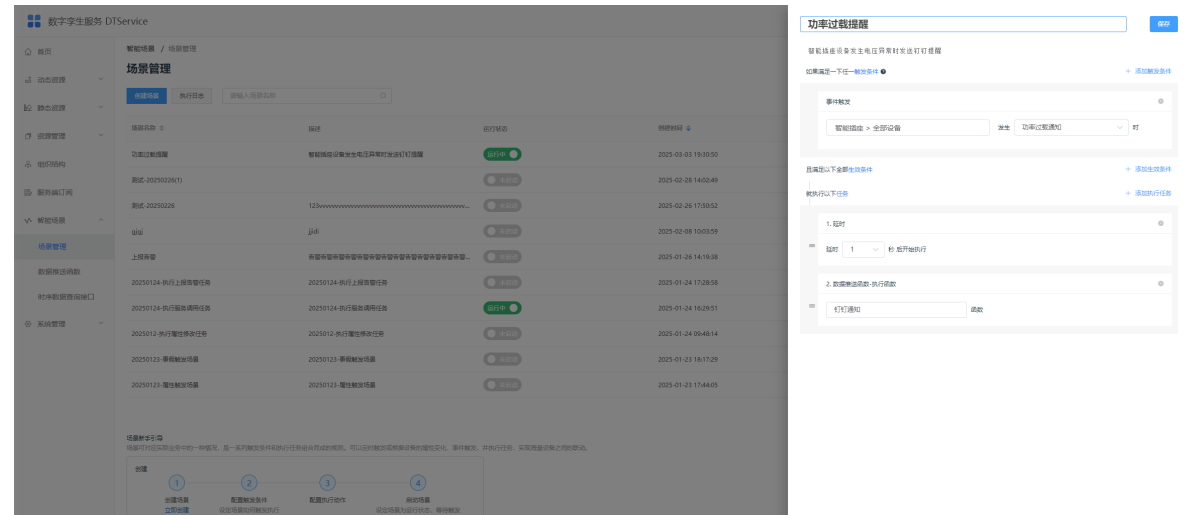
创建任务

在任务管理页面，创建一个任务名为“功率过载提醒”的任务。

任务逻辑

智能插座设备发生功率过载事件 触发 执行任务 发送消息到钉钉通知

创建完成后，进入编辑页面。根据任务逻辑编辑对应的规则。设定触发条件为事件触发：智能电压设备发生功率过载事件，添加执行任务类型为“数据推送函数”，选择数据推送函数“钉钉通知”。编辑完成后保存并开启任务。



第三步：配置一条超温告警规则



模拟事件触发

使用APIFox模拟功率过载事件的触发。

结果验证

事件触发后，前往钉钉群查看消息通知。



第四步：在低代码设计器中绑定并呈现

DTS数据孪生架构提供快速接入查看DTS实体数据的平台功能[低代码设计器](#)，详细可视化功能可查看《[BigView低代码设计器教程](#)》。另一种方式是经过[订阅管理](#)将向DTS订阅实体的属性变化和事件发生的数据，订阅端收到数据之后自行进行可视化处理便能对设备进行实时数据监控。示例以查看温度传感器的属性数据为例。

1. 登录低代码设计器；
2. 进入数据集模块，选择新建一个“实体属性数据集”；
3. 输入数据集名称为“温湿度传感器属性数据集”，选择温湿度传感器—>温湿度传感器1751884372作为实体对象，获取此实体的属性数据；
4. 点击解析并运行，在数据预览区查看请求结果；
5. 编写脚本，从请求结果中获取需要的数据内容；

示例脚本

```
const input=resp.info;
// 提取并重组数据
const resultArray = input.payload.map(item => ({
  deviceId: input.deviceId, // 从顶层取deviceId
  deviceName: input.deviceName, // 从顶层取deviceName
  propertyId: item.propertyId,
  propertyIdentifier: item.propertyIdentifier,
  propertyName: item.propertyName,
  propertyValue: item.propertyValue
}));
return resultArray
```

示例数据预览效果

deviceId	deviceName	propertyId	propertyIdentifier	propertyName	propertyValue
40756a2c-1880-4d6c-a28a-c106c238016c	温湿度传感器1751884372	f5345a6c90864715	temperature	当前温度	32.19
40756a2c-1880-4d6c-a28a-c106c238016c	温湿度传感器1751884372	ea9f30ac3930a049	humidity	当前湿度	6.42
40756a2c-1880-4d6c-a28a-c106c238016c	温湿度传感器1751884372	f2884913bdc641ab	batteryLevel	电量信息	33

1. 保存数据集。
2. 进入布局管理模块，选择新建一个布局，输入名称为“温湿度传感器”；
3. 布局创建完成后，选择设计布局；
4. 在设计器中，拖拽文本组件到画布中；
5. 设置文本组件的数据集为“温湿度传感器属性数据集”，显示字段 'DeviceName'
6. 拖拽多个文本组件用于显示温湿度和电量属性，分别设置文本组件的数据源为“表达式”，使用表达式从数据集中获取数据。

示例表达式

```
return dataset['温湿度传感器_ZKzhDxZG'].find(item => item.propertyName === "当前温度").propertyValue + "°C"
```



第四步：在低代码设计器中绑定并呈现



操作指南

操作指南

概念定义

概念定义

实体管理

名词	描述
实体管理	数字孪生服务中的产品、实体、网关的管理。
产品	一组具有相同功能的实体的集合，产品定义了实体的标准属性、事件、服务，可以理解为某类实体的本体描述。 DTS为每个产品颁发全局唯一的productId。 每个产品下最多可以包含50万个实体。
实体	产品的实例化对象，是某个产品下的具体实体， 可以来源于物联网或虚拟世界应用。DTS为实体颁发、 自定义或导入唯一标识deviceId，一般由字母、数字和特殊字符组成， 具有配套密钥deviceSecret，由DTS配套生成，一般为8- 32个16进制字符。实体需要通过网关连接DTS。
组合实体	一种特殊的虚拟实体，是由多个实体根据应用需要组合的一种实体， 其下的每个实体都是一个原实体的镜像， 也就是组合实体无需为每一个实体新建产品Profile。
所属品类	创建产品时选择的功能模板分类。包括 标准品类 （使用系统预先配置好常用功能模板的产品，可一键导入）与 自定义品类 （不带任何预设模板，需用户完全自定义属性、事件与服务）。
接入协议	物理或虚拟设备与网关通信所使用的通信网络协议类型。 包括通用网关自带的 HTTP 协议 （提供设备接入、属性/事件上报）， 以及对于非标设备接入需专门定制开发对应的 定制协议 接入服务。
数据格式	设备与平台交互的数据序列化结构形式。包括平台标准的 BV JSON （采用预定义 JSON 结构直接通信）与设备自定义的 透传/自定义 格式 （后者需要消息解析脚本在平台中进行上行与下行格式转换）。
网关	网关是 DTS 进行设备数据接入与协议解析的核心通道。 每一个创建的网关都必须对应底层服务器上实际运行的协议接收服务（如 HTTP、MQTT 监听程序或容器服务等），不能随意空置。网关负责聚合/ 代理下属子实体的数据流，执行接入鉴权，并处理非标协议解析。
	系统内置的逻辑虚拟网关

名词 Dumb网关	描述需要任何底层的物理网络通道或协议接收服务。 专门用于管理不具备物理通信实体（即无需网络连接）的“组合实体”虚拟映射，以维持数字孪生模型层次结构的完整性。
--------------	---

物联网网关	依托底层特定的数据通信协议（如 HTTP、MQTT 等）接收服务运行的网关通道。 专门用于代理挂载在该协议下的多个物理子实体接入 DTS，完成数据的集中上报与指令下发。
视频网关	依托底层的视讯/流媒体处理服务（如系统内置的视频流媒体中间件，如内置的 BVMedia 服务）运行的网关通道。 专门用于视频实体设备流数据源与状态监控的接入和关联管理。
实体证书	实体证书指 <code>productId</code> 、 <code>deviceId</code> 、 <code>deviceSecret</code> 。 productId ：是DTS为产品颁发的全局唯一标识。该参数很重要，在实体认证以及通信中都会用到。 deviceId ：在注册实体时，自定义的、系统生成的或从数字孪生服务导入的实体名称，具备产品维度内的唯一性。该参数很重要，在实体认证以及通信中都会用到。 deviceSecret ：DTS为实体颁发的实体密钥，和deviceId成对出现。该参数很重要，在实体认证时会用到。
Topic	Topic是UTF-8字符串，是发布（Pub）/订阅（Sub）消息的传输中介。可以向Topic发布或者订阅消息。
Topic类	同一产品下不同实体的Topic集合，用 <code>\${productId}</code> 和 <code>\${deviceId}</code> 通配一个唯一的实体，一个Topic类对一个productId下所有实体通用。
发布	操作Topic的权限类型，对应的英文名称为Pub。 可以往此类Topic中发布消息。
订阅	操作Topic的权限类型，对应的英文名称为Sub。 可以从此类Topic中订阅消息。
标签	标签用于对平台资源进行分类检索与授权隔离。标签采用二级格式：一级为“标签分组”（如“业务线”、“部署区域”），二级为具体“标签”（如“制造部”、“北京中心”）。标签编辑与筛选全面支持“可选择 + 可输入”的组合模式（既能快速选择已有标签，也允许直接输入新建）。在进行标签筛选检索时，默认匹配该标签分组下的全部二级标签。目前支持标签的模块包括：产品、网关、实体、组合实体、资源、资源组、任务、数据推送函数、时序数据记录

名词	描述
产品Profile	对实体的功能描述，包括实体的属性、服务和事件。 BV数字孪生模块通过定义一种物的描述语言来描述产品模型，称之为Profile，采用JSON格式，可以根据Profile组装上报实体的数据。

属性	对一个具体状态量、参数量或可配置量的定义，例如温度、速度、位置、模式。 例如对于独立的温度和湿度测量时序数据，可以用一个“温湿度传感器”实体的“温度”、“湿度”两个属性分别进行定义，实现实体属性的正向映射。属性有两种更新类型： 遥测指标（默认） ：动态或快变数据的更新方式，对数据消费者来说，这些传感器的测量值仅实时有意义，过时的数据就没用了，同时它们的更新频率往往很高，所以默认不存入数据库，仅每个心跳周期内存储第一条数据。对于历史数据流的诊断分析应用，可以选择将其进行时序数据记录，存入专门的时序数据库； 实体属性 ：静态或慢变属性的更新方式，如设备的固件版本、设备名称，它们一般很少变化，每次属性更新都会存储到数据库。
事件	实体主动发出的、具有时间点意义的事实，例如告警、碰撞、进入区域、任务完成、状态异常。事件是包含事实描述的通知信息，可包含多个输出参数。事件可以被订阅和推送。
服务	外部可对实体发起的操作，实体以服务的方式提供，例如启动设备、切换模式、打开舱门、执行仿真动作。 实体的服务是对反向控制接口调用或能力的封装。
消息解析脚本	针对采用透传格式/自定义数据格式的实体，需要在DTS编写数据解析脚本，将实体上报的二进制数据或自定义的JSON数据，转换为DTS支持的JSON数据格式；将平台下发的JSON格式数据，转换为实体支持的格式。

分组管理

名词	描述
资源	资源指数字孪生服务资源，即数字孪生服务上的产品、实体、网关等内容。
资源组	资源组是数字孪生服务资源的分组。通过资源分组管理，可以授予不同账号查看和操作具体数字孪生服务资源（产品、实体和任务）的权限，从而实现不同账号之间的资源隔离

组织结构

名词	描述
数字孪生模型	物理世界的人员、环境、实体及虚拟世界在数字空间中的全面呈现。在 DTS 中，主要基于物理实体与孪生节点构建，表现为一棵支持层级管理的组织结构树；而涉及三维渲染、业务应用和高级渲染层的虚拟世界模型，则在 DTC 中进行托管与应用。
数字孪生图	呈现项目空间、实体、人员之间多维关联关系的网状拓扑图。在 DTS 侧，尽管配置上表现为树状菜单，但由于 孪生体和实体节点支持复用（即一个节点可以同时挂载到多个上级节点） ，其底层逻辑构成了一张多对多关联的网状图，为 DTC 进行跨维度数据聚合和联动提供支持。
组织结构	数字孪生系统中对孪生体、实体（设备）及关联人员等要素之间层级与关联关系的逻辑框架管理。用户可借此描述空间划分（如厂区-车间-产线-设备）、行政隶属关系或业务逻辑分类，是平台实现数据聚合呈现、策略下发和权限隔离的基础。
根节点Root	系统内置的最顶层逻辑节点，是整个组织结构树的起始原点。所有的项目孪生体、DTS 内部结构以及自定义孪生体都作为根节点的子树向下延伸。该节点在系统中不可删除或重命名。
DTS内部结构	系统内置的独立组织结构根节点。与绑定到特定 DTC 项目的“项目孪生体”不同，DTS 内部结构是平台全局独立的虚拟孪生空间，用户在此处构建的树状结构不依赖于具体 DTC 业务项目，仅供 DTS 进行全局资产管理、跨项目共享或通用调试。
项目孪生体	来自于综合应用管理平台 DTC，以项目为最上层（根节点）的孪生体结构，作为该项目下所有物理和虚拟资源的顶层容器。在 DTS 中，用户可以 编辑和修改其子级节点结构及实体的挂载关系 ，但不支持在 DTS 中创建或删除项目孪生体本身 （生命周期由 DTC 侧统一管理）。
自定义孪生体	自由度最高的逻辑孪生体类型。完全由用户在 DTS 中根据业务需求（如空间逻辑划分）自主创建。用户可以 自由构建多级树状结构、重命名节点，并随时添加、移动或删除关联的实体和下级节点 。
结构孪生体	基于 DTC “ 结构孪生体模板 ”实例化的孪生体。在 DTS 中其 功能完全受限 ，不仅 不可编辑其节点树结构 ，也无法在 DTS 中绑定或解绑具体的实体 （所有结构与实体关联关系的修改均必须在 DTC

名词	描述
	通过其模板统一进行）。DTS 不支持该类型孪生体的创建。
单孪生体	基于 DTC “ 单孪生体模板 ” 创建的实例化孪生体（支持嵌套及拥有子孪生体，例如“标准楼层”模板）。该类型孪生体 可通过模板对树状层级结构进行重复利用与统一修改 ；同时， 各个实例化的单孪生体下所绑定的具体实体是可以不同且独立配置的 。DTS 不支持该类型孪生体的创建。

任务管理

名词	描述
任务	自动化业务逻辑，在满足条件下后，数字孪生服务会自动执行预设的任务列表。
触发条件	用以触发任务开始执行的条件。包含定时触发、属性触发、事件触发； 定时触发 ：到达设定的时间点时触发执行，可以设定重复的周期； 属性触发 ：指定实体发生属性上报且满足设定的属性状态时触发； 事件触发 ：指定实体发生指定事件时触发；
生效条件	在任务满足触发条件后需要满足的附加生效条件。需全部满足生效条件任务才会被执行
执行任务	任务执行时需要执行的任务列表。属性修改：直接修改实体的属性值。服务调用：调用实体的服务以达到某个功能。执行任务：触发执行任务启动/停止任务：修改任务的运行状态。数据推送函数：执行数据推送函数。延时：延时一定时间后继续执行下一个任务。 注意 以触发事件输出参数作为参数值输入到服务调用或上报告警内容时，需触发条件含有事件触发，否则参数赋值会出现异常。
数据推送函数	提供灵活便捷的数据交互转发模式，可在任务中作为任务被执行，支持RedisMQ、Http请求以及时序数据记录。
时序数据记录	将DTS接收的属性上报、事件上报根据需求选取所需字段内容一条条记录到数据库中，作为后续使用的数据记录
时序数据查询接口	将存储的属性和事件信息数据对外提供访问的途径。

订阅管理

名词	描述
订阅	订阅产品发生变更的业务信息（如实体数据上报、事件数据上报、属性数据上报、实体状态变化等）。订阅一个或多个事件并指定操作类型、订阅者以及消息质量。数字孪生服务根据设置的订阅，将产品下所选择实体的该事件消息流转至目标订阅者。
订阅者	向数字孪生服务发起订阅的第三方服务端或应用平台。
消费者	消费者指在BV数字孪生服务为订阅应用端创建的扩展体系。在消费者扩展体系内可以创建组织架构、用户组、用户并为用户指定角色。订阅者可以根据服务端或应用平台内部的用户体系创建消费者，配置订阅关系后，DTS会将资源下所有实体或POI的已订阅类型的消息转发至指定的消费者。

实体管理

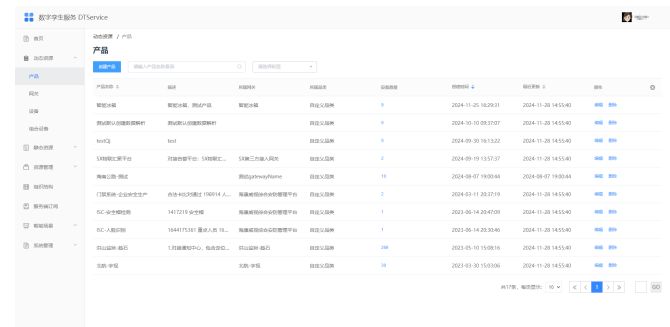
实体管理

产品

产品

通过本文档你可以了解到产品相关的基础操作。

产品是实体的集合，通常是一组具有相同功能定义的实体集合。



创建产品

选中【产品】选项卡，在页面下选择【创建产品】，进入创建产品页面。输入新建产品的信息后，点击【立即创建】。

![./.../Pasted image 20260623102412.png]]

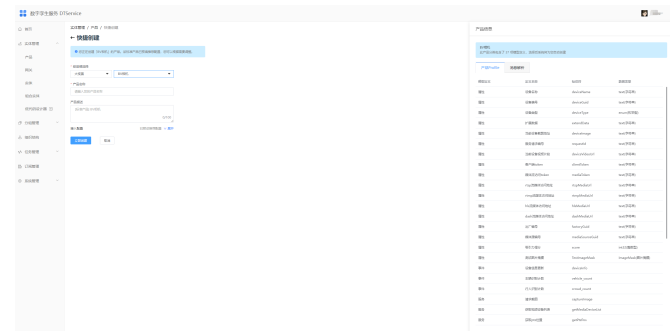
参数	描述
产品名称	为产品命名。 产品名称具有唯一性。例如，可以填写为产 品型号。 支持中文、英文字母、数字、下划线（_）、连接号（-）、@符号和英文圆括号， 长度限制4~30， 一个中文汉字算2位。
接入网关协议	物理/ 虚拟设备接入网关所使用的网络协议，可选“HTTP协议”与“定制协议”， 详细定义与区别请参考 概念定义 - 接入协议 。

参数	描述
数据格式	设备与平台数据交互的格式，可选“BV JSON数据格式”与“透传/自定义”，详细定义与区别请参考 概念定义 - 数据格式 。
认证方式	实体接入DTS的安全认证方式。产品创建成功后，认证方式不可变更。可选：实体密钥：使用DTS为实体生成的deviceSecret进行实体认证签名计算。
产品描述	可输入文字，用来描述产品信息。字数限制为100。

快捷创建

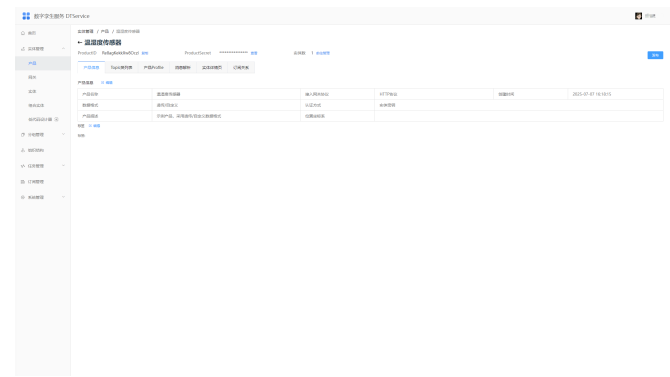
1. 选中【产品】选项卡，在页面下选择【快捷创建】，进入快捷创建产品页面。
2. 选择已有的品牌下的产品，通过已有产品的模板进行快速创建。
3. 输入产品名称，点击【立即创建】。
4. 产品的接入协议、数据格式、认证方式、坐标系、关联网关默认与预设的产品配置一致，可自定义修改。

快捷创建的产品会根据选择产品自动生成产品Profile和消息解析脚本。



产品详情

在产品页面选择需要编辑查看的产品，点击“编辑”进入产品详情页面。在产品详情页展示所有与产品相关的数据内容，可查看或编辑 [产品基本信息](#)、[Topic类](#)以及[产品Profile](#)，[消息解析脚本](#)、[实体详情页](#)、[订阅关系](#)等内容。



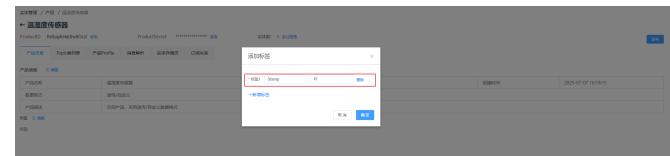
产品基本信息

可修改产品的名称以及产品描述等基本信息，但产品的接入信息如接入网关协议等内容不可重新编辑修改。



设置产品标签

产品标签用于对产品以及该产品下所有实体所具有的共性属性和业务属性进行逻辑分组（如产品的制造商、所属区域等）。产品标签支持“可选择 + 可输入”的组合模式，既可以从下拉列表中快速选择已有的分组或标签，也允许自定义输入新标签。



参数	说明
标签分组 (一级标签)	标签的类别或属性组，支持选择已有分组或手动输入创建（如 Manufacturer 、 Region ）。
标签 (二级标签)	具体的标签值，支持选择已有标签或手动输入创建（如 大视景 、 华东区 ）。

Topic类列表

产品Topic类

Topic类是一类Topic的集合。

产品

创建实体后，产品的所有Topic类会自动映射到实体上，无需单独为每个实体创建Topic。

通信Topic

产品的Topic类不用于通信，只是定义Topic。用于消息通信的是具体的实体Topic。



系统Topic

DTS预定义的Topic。

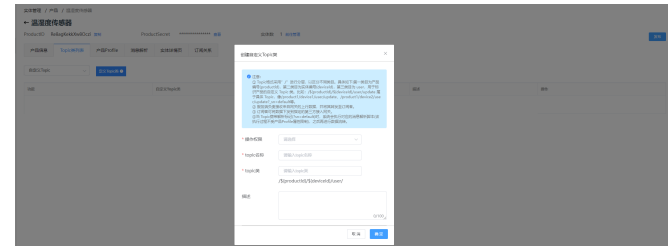
系统Topic包含展示在控制台产品、实体详情页下的Topic和各功能使用的Topic。



自定义Topic

自定义Topic通信不使用产品模型profile，消息的数据结构由网关服务根据业务自定义。自定义Topic主要解决产品关系复杂，数据关联较多，无法通过产品Profile很好的体现。比如大数据量的资源传输、产品模型结构复杂、业务偏向于资源管理无事件服务、添加产品Profile操作繁琐等情况下。

定义Topic类



参数	描述
实体操作权限	实体对该Topic的操作权限，可设置为 发布 、 订阅 。
Topic类	根据实体Topic规则，设置Topic类的自定义类目名称
描述	对自定义的Topic类进行描述，可以为空。

产品Profile

产品Profile描述产品“是什么，能做什么”，构建该实体的数字化数据模型。完成功能定义后，系统将自动生成该产品模型。
当定义完一款产品Profile后，在创建实体时，就可以选择已经创建好的该产品Profile。

[!NOTE]
关于产品Profile的基础概念，请参考[概念定义 - 产品Profile](#)。

选择【编辑草稿】进入产品Profile的编辑状态。在完成变成编辑后，需点击【发布】将编辑内容保存。

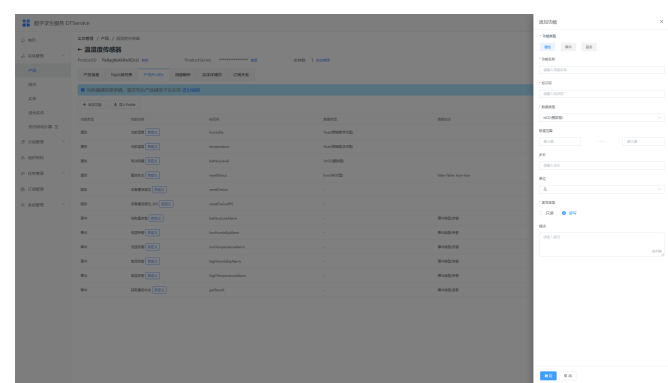
添加产品profile

可为产品添加属性、服务、事件，且各产品Profile参数皆为自定义类型。

添加属性

用于添加和配置描述实体基础信息或运行时状态的属性字段。

[!NOTE]
关于属性与遥测指标的分类定义，请参考[概念定义 - 属性](#)。



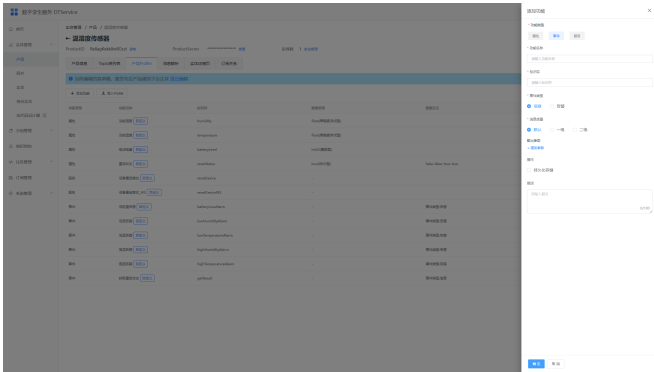
参数	描述
功能名称	属性的名称，同一产品下功能名称不能重复。
	说明： 支持中文、大小写字母、数字、短划线和下划线，且必须以中文、英文或数字开头，不超过30个字符。
标识符	属性唯一标识符，在产品中具有唯一性。 即JSON格式中的 identifier 的值， 作为实体上报该属性数据的Key， DTS根据该标识符校验是否接收数据。可包含英文、 数字、下划线，长度不超过50个字符， 例如PowerConsumption。

参数	描述
	说明 不能用以下系统保留参数作为标识符：set、get、post、time、value。
数据类型	<p>- int32：32位整型。需定义取值范围、步长和单位符号。 - float：单精度浮点型。需定义取值范围、步长和单位符号。 - double：双精度浮点型。需定义取值范围、步长和单位符号。 - enum：枚举型。</p> <p>定义枚举项的参数值和参数描述，例如1-加热模式、2-制冷模式等。 - bool：布尔型。采用0或1来定义布尔值，例如0-关；1-开。 - text：字符串。</p> <p>需定义字符串的数据长度，最长支持2048字节。 date：时间戳。格式为String类型的UTC时间戳，单位：毫秒。 - struct：JSON对象。定义一个JSON结构体，新增JSON参数项，例如定义灯的颜色是由Red、Green、Blue三个参数组成的结构体。不支持结构体嵌套。 - array：数组。需声明数组内元素的数据类型，可选择int32、float、double、text或struct。</p> <p>需确保同一个数组元素类型相同。数组内可包含1-128个元素。</p>
步长	<p>属性值和事件以及服务中输入输出参数值变化的最小粒度。数据类型为int32、float、double时，根据业务需要设置步长。</p> <p>例如为温度计产品定义温度属性时，将数据类型设置为int32，步长为2，单位为℃，取值范围0~100。即温度每变化两度，实体上报温度值，例如0℃、2℃、4℃、6℃、8℃.....。</p>
单位	单位可选择为无或根据实际情况选择。
读写类型	<p>- 读写：请求读写的方法支持GET（获取）和SET（设置）。 - 只读：请求只读的方法仅支持GET（获取）。</p> <p>说明 属性可以通过方法上报，如果部分属性提供设置，可选择读写类型，数字孪生将提供设置接口，用来接收订阅者发来的设置属性信息。</p>
描述	输入文字，对该功能进行说明或备注。长度限制为100字。

添加事件

用于定义实体运行时的通知或告警事件，支持配置多个输出参数。

[!NOTE]
关于事件的作用与分类定义，请参考[概念定义 - 事件](#)。



参数	描述
功能名称	事件的名称，同一产品下功能名称不能重复。 说明 支持中文、大小写字母、数字、短划线和下划线，且必须以中文、英文或数字开头，不超过30个字符。
标识符	事件唯一标识符，在产品下具有唯一性。 即JSON格式中该事件的 identifier 的值， 作为实体上报该事件数据的Key，例如 ErrorCode。 说明 不能用以下系统保留参数作为标识符：set、get、post、time、value。
事件类型	- 信息：指实体上报的一般性通知，例如完成某项任务等。 - 告警：实体运行过程中主动上报的突发或异常情况， 告警类信息，优先级高。 可以针对不同的事件类型进行业务逻辑处理和统计分析。 说明 信息事件类型不会进行通知中心以及系统弹框提醒。
消息质量	上行数据流转的消息质量。 说明 默认代表qos=0； 一级代表qos=1；二级代表qos=2；
持久化存储	是否持久化存储。 说明 消息通过透传的方式进行持久存储用户模板渲染以及历史追溯。 属性数据默认存储采用更新的方式存储。
输出参数	事件的出参。 自定义参数。 说明 - 不能用以下系统保留参数作为输出参数的标识符：set、get、 post、time、value。 - 一个事件最多支持定义50个出参。

参数	描述
输出模板	输出参数展示样式，对外响应效果。看板： 选择看板作为事件展示时的模板样式， 可通过低代码设计器绘制看板。 说明： 看板将用来实时渲染展示， 请确保输出参数 identifier 与看板使用的数据集 identifier 参数一致。不添加模板则不会进行页面渲染直接发送Json数据， 默认发送模板链接以及json数据；
描述	输入文字，对该事件功能进行说明或备注。长度限制为100字。

事件模板

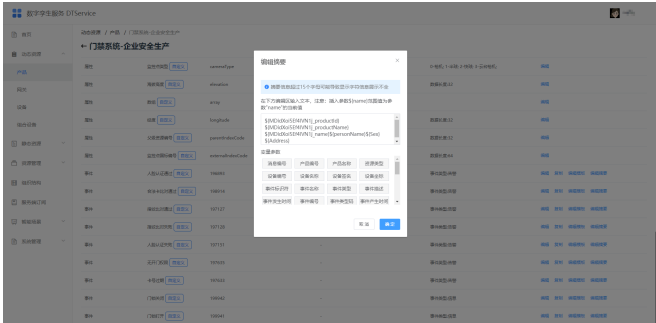
事件模板是展示**事件**数据的一个可视化页面，可以展示以及渲染到第三方平台，进行静态页面展示。选择看板作为事件展示时的模板样式，可通过低代码设计器绘制看板。将选择的看板保存后就成为该事件的模板。

说明：将用来实时渲染展示，请确保输出参数**identifier**与看板使用的数据集**identifier**参数一致。不添加模板则不会进行页面渲染直接发送Json数据，默认发送模板链接以及json数据；



事件摘要

事件摘要是通过文本的形式简要明了的描述发生的事件。可以自行输入文本内容，也可以通过事件输出参数拼装组成。在发送事件摘要到通知中心时，会将数据值装载到摘要中。



添加服务

用于定义实体可被外部调用的能力或方法，支持配置输入与输出参数。

[!NOTE]
关于服务的机制与功能定义，请参考[概念定义 - 服务](#)。

服务地址和接口支持**动态参数**定义，输入格式为\${参数名}（例如：/\${admin}）。

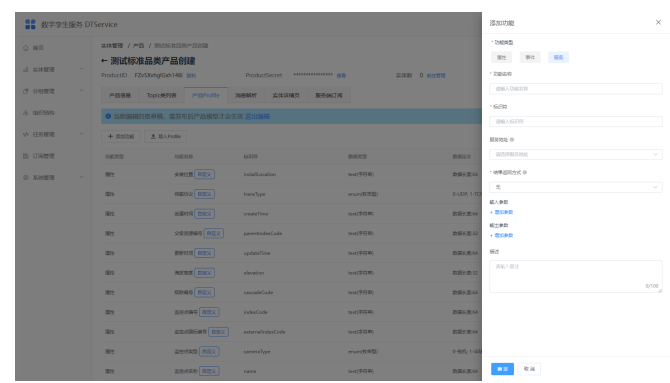
若配置了动态参数，需在产品服务定义下“输入参数”列表中定义同名的参数（如admin）。实际请求时，系统将自动使用该参数的值替换地址中的动态部分，生成完整的请求URL。

示例说明：

接口地址：[http://api.example.com/\\${admin}/status](http://api.example.com/${admin}/status)

输入参数中定义：admin（值为visitor）

最终请求URL：<http://api.example.com/visitor/status>

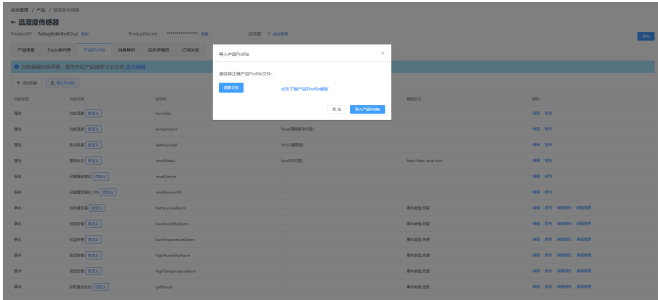


参数	描述
功能名称	服务名称，同一产品下功能名称不能重复。 说明 支持中文、大小写字母、数字、短划线和下划线，且必须以中文、英文或数字开头，不超过30个字符。
标识符	服务唯一标识符，在产品下具有唯一性。即JSON格式中该服务的 identifier 的值。可包含英文、数字、和下划线，长度不超过30个字符。 说明 不能用以下系统保留参数作为标识符：set、get、post、time、value。
服务地址	通过产品Profile服务调用外部服务的请求目标地址。服务地址需在产品关联的网关下进行维护。目前支持三种调用方式：HTTP方式、RedisMQ方式、WebSocket方式。其中WebSocket方式适用于局域网内与第三方平台的通信。 说明 只有HTTP方式支持同步和异步的调用选择，其余方式只能使用异步调用。

参数	描述
结果返回方式	服务执行完成后， 是否需要将服务执行结果反馈到DTS。 - 属性上报： 服务提供方将执行结果以属性上报的形式将结果返回； - 事件上报： 服务提供方将执行结果以事件的形式将结果返回； - HTTP回调：通过HTTP将结果返回， 需输入结果返回地址， 服务提供方需将结果转发至目标地址； - Redis回调： 通过Redis将结果返回，需输入结果返回地址， 服务提供方需将结果转发至目标地址。 说明 属性上报和事件上报会将结果在DTS内部进行流转， HTTP和Redis方式服务提供方将结果转发至目标地址， 数据不会回到DTS。
输入参数	设置该服务的入参，可选。 说明 不能用以下系统保留参数作为输入参数的标识符： set、get、post、time、value。可以自定义参数。 一个服务最多支持定义 20 个入参。
输出参数	设置该服务的出参，可选。 说明 不能用以下系统保留参数作为输出参数的标识符： set、get、post、time、value。可以自定义参数， 一个服务最多支持定义20个出参。
描述	输入文字，对该服务功能进行说明或备注。 长度限制为100字。

导入产品profile

编辑产品Profile的草稿，选择【导入Profile】，上传产品ProfileJSON文件（具体文件格式请查看产品Profile格式），即可快速添加一个产品Profile。



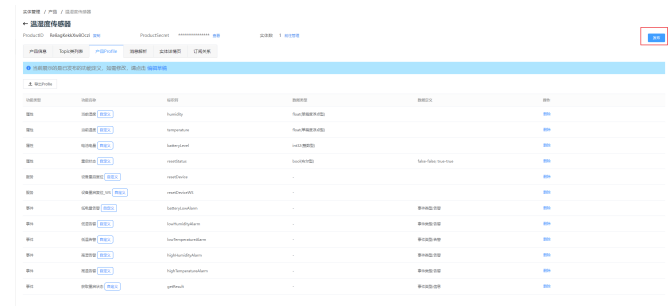
注意：导入产品Profile后，会覆盖该产品原有的功能定义。事件模板不能批量添加。

导出产品profile

编辑完成的产品Profile可以导出为Json文件形式，方便产品的转移。

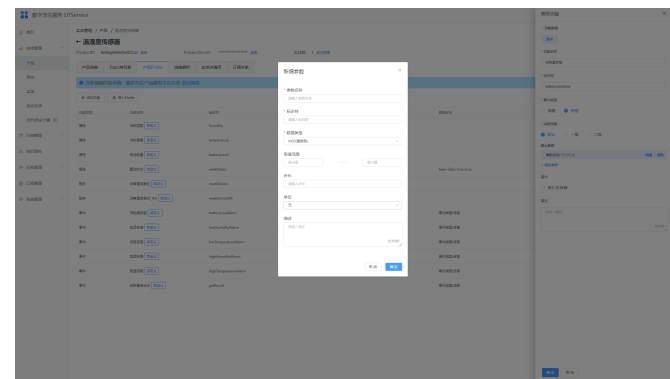
发布产品profile

定义或修改产品Profile之后都需要将新的产品Profile进行发布。发布之后的产品Profile才能被DTS所识别。



输入/输出参数

定义事件以及服务中输入输出参数值。



参数	描述
参数名称	参数的名称。 说明 支持中文、大小写字母、数字、短划线和下划线，且必须以中文、英文或数字开头，不超过30个字符。
标识符	参数唯一标识符。作为服务以及事件输出输入参数的Key，数据 孪生模块根据该标识符校验是否接收数据以及模板渲染。 可包含英文、数字、下划线，长度不超过50个字符，例如PowerConsumption。 说明 不能用以下系统保留参数作为标识符：set、get、post、time、value。

参数	描述
数据类型	<p>- int32：32位整型。需定义取值范围、步长和单位符号。 - float：单精度浮点型。需定义取值范围、步长和单位符号。 - double：双精度浮点型。需定义取值范围、步长和单位符号。 - enum：枚举型。定义枚举项的参数值和参数描述，例如1-加热模式、2-制冷模式等。 - bool：布尔型。采用0或1来定义布尔值，例如0-关；1-开。 - text：字符串。需定义字符串的数据长度，最长支持2048字节。 - date：时间戳。格式为String类型的UTC时间戳，单位：毫秒。 - struct：JSON对象。定义一个JSON结构体，新增JSON参数项，例如定义灯的颜色是由Red、Green、Blue三个参数组成的结构体。不支持结构体嵌套。 - array：数组。需声明数组内元素的数据类型，可选择int32、float、double、text或struct。需确保同一个数组元素类型相同。数组内可包含1-128个元素。</p>
步长	<p>属性值和事件以及服务中输入输出参数值变化的最小粒度。数据类型为int32、float、double时，根据业务需要设置步长。例如为温度计产品定义温度属性时，将数据类型设置为int32，步长为2，单位为℃，取值范围0~100。即温度每变化两度，实体上报温度值，例如0℃、2℃、4℃、6℃、8℃……。</p>
单位	单位可选择为无或根据实际情况选择。
读写类型	<p>- 读写：请求读写的方法支持GET（获取）和SET（设置）。 - 只读：请求只读的方法仅支持GET（获取）。</p> <p>说明 属性可以通过方法上报，如果部分属性提供设置，可选择读写类型，数字孪生将提供设置接口，用来接收订阅者发来的设置属性信息。</p>
描述	输入文字，对该功能进行说明或备注。长度限制为100字。

消息解析

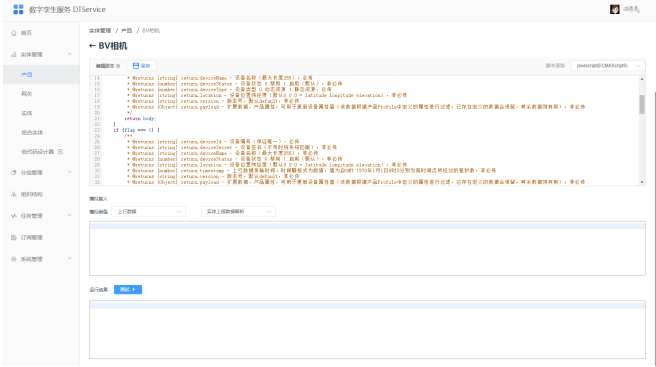
数据格式为透传/自定义的产品下的实体与DTS进行数据通信时，需要DTS调用定义的消息解析脚本，将上、下行产品Profile数据分别解析为DTS的标准格式（BV JSON）和实体的自定义数据格式。

DTS接收到来自实体的数据时，先运行解析脚本，将透传的数据转换成BV JSON格式的数据，再进行业务处理；DTS下发数据给实体前，也

会先通过脚本将BV JSON格式的数据转换为实体的自定义格式，再下发给实体。

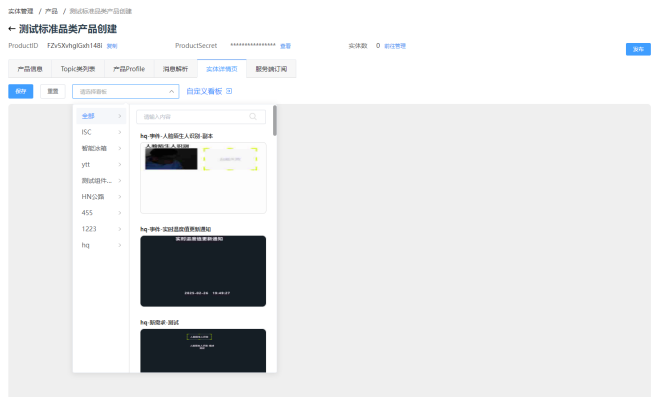
在编辑脚本框内输入消息解析的脚本，可在下方模拟输入上行或下行数据测试脚本是否能够正常运行。

建议：事先在开发工具中编写测试好对应的脚本后，在移植在数字孪生服务中，进行简单修改即可。



实体详情页

实体详情页用于展示以及渲染到应用程序(MRC/ARC等)，进行静态页面展示。在应用程序在选择实体后，会自动弹出详情页。看板的创建需要通过低代码设计器进行。如何创建看板可参考[低代码设计器](#)。



订阅关系

当前产品存在订阅时，可查看所有订阅的信息。



网关

网关

网关是 DTS 进行物理/虚拟设备数据接入与协议解析的核心通道。**网关在平台上不能随意配置；每一个网关的创建，都必须对应底层服务器上实际运行的接入协议服务（如 HTTP、MQTT 等网络监听服务）。不同的通信协议和数据类型必须使用相匹配的网关进行数据接入。**

DTS 提供了三种类型的网关：物联网网关、视频网关以及专门管理组合实体的虚拟 Dumb 网关。

[!NOTE] 关于各类网关的技术机理与协议支撑定义，请参考[概念定义 - 网关](#)。

创建网关

数字孪生服务 DTService

首页

动态资源

产品

网关

设备

组合设备

静态资源

资源管理

组织结构

服务订阅

智能场景

系统管理

动态资源 / 网关 / 创建网关

← 创建网关

* 网关名称

请输入网关名称

* 标识符

请输入标识符

* 网关类型

Dumb网关(默认)

关联产品

已选择0个产品

网关属性

+ 新增属性

网关描述

请输入网关描述 0/100

立即创建

取消

[!IMPORTANT] 网关配置完成后，设备或数据采集程序需要使用该网关的证书（ `GatewayKey` 与 `GatewaySecret` ）连接服务器上对应的协议接收服务。只有该协议接收服务处于正常运行状态且设备鉴权通过，实体的数据流才能成功流入 DTS。

参数	描述
网关名称	为网关命名。网关名称具有唯一性。支持中文、英文字母、数字、下划线（_）、连接号（-）、@符号和英文圆括号，长度限制4~30，一个中文汉字算2位。
标识符	网关唯一标识符，在产品下具有唯一性。可包含英文、数字、和下划线，长度不超过30个字符。 说明 不能用以下系统保留参数作为标识符：set、get、post、time、value。

参数	描述
网关类型	网关类型，Dumb网关、视频网关、物联网网关。
关联产品	网关关联的产品，产品下的实体都挂载在网关下。
网关属性	实体的公共属性，可自定义属性。 说明 不能用以下系统保留参数作为公共参数的标识符：set、get、post、time、value。一个网关公共参数最多支持定义20个参数。
网关描述	可输入文字，用来描述网关信息。字数限制为100。

平台已内置名为“通用网关”的网关，可支持加入BV协议接入、第三方协议接入（仅Http协议）的产品。内置“BVMedia”网关可支持从看门狗服务中接入相机产品。

编辑网关

查看网关信息，可以对网关的[基本信息](#)、[网关属性](#)、[服务列表](#)、[发现实体接口](#)进行编辑修改。

实体管理 / 网关 / BVMedia

← BVMedia

GateWayKey gateway00007 复制 GateWaySecret ***** 查看 实体数 70 前往管理

网关信息 网关属性 服务列表 发现实体接口

网关编号	76441b8484664921	网关名称	BVMedia	标识符	BVMedia
网关类型	视频网关	创建时间	2024-07-25 16:16:52	最近更新	2025-03-19 09:15:30
关联产品	BV相机, bv相机2				
网关描述	大视景看门狗服务				

标签 编辑

标签:

修改网关信息

可以对网关的名称、网关类型、关联产品、网关描述进行修改，如增加或减少网关下关联的产品。

← BVMedia

GateWayKey gateway00007 复制 GateWaySecret ***** 查看 实体数 70 前往管理

网关信息 网关属性 服务列表 发现实体接口

网关信息 编辑

网关编号	76441b8484664921	网关名称	BVMedia	标识符	BVMedia
网关类型	视频网关	创建时间	2024-07-25 16:16:52	最近更新	2025-03-19 09:15:30
关联产品	BV相机, bv相机2				
网关描述	大视景看门狗服务				

标签 编辑

标签:

网关标签

用于标注同一个网关下所有实体所具有的共性物理或逻辑环境信息（如网关部署位置、通信环境等）。网关标签支持“可选择 + 可输入”的组合模式，既支持从已有列表中快速选择，也允许自定义输入。

[!NOTE] 关于网关标签的具体定义，请参考[概念定义 - 标签](#)。

参数	说明
标签分组 (一级标签)	标签的类别或属性组，支持选择已有分组或手动输入（如 <code>Deployment</code> 、 <code>Location</code> ）。
标签 (二级标签)	具体的标签值，支持选择已有标签或手动输入（如 <code>Production</code> 、 <code>Room-102</code> ）。

网关属性

使用网关属性管理与网关相关的属性信息、产品通信时的通用信息，如IP地址、端口等内容。网关属性添加后可以进行属性赋值，赋值后的网关属性就可以作为**参数值**在任务等功能中使用。

实体管理 / 网关 / BVMedia

← BVMedia

GateWayKey gateway00007 复制 GateWaySecret ***** 查看 实体数 70 前往管理

网关信息 网关属性 服务列表 发现实体接口

+ 添加属性 属性赋值

功能类型	功能名称	标识符	数据类型	数据定义	属性值	操作
属性	SERVER_IP	SERVER_IP	text(字符串)	数据长度:128	192.168.100.172	编辑 删除
属性	SERVER_PORT	SERVER_PORT	text(字符串)	数据长度:128	14853	编辑 删除
属性	TIME_OUT	TIME_OUT	text(字符串)	数据长度:128	10000	编辑 删除
属性	USER_NAME	USER_NAME	text(字符串)	数据长度:128	admin	编辑 删除
属性	PASSWORD	PASSWORD	text(字符串)	数据长度:128	admin	编辑 删除
属性	SERVICE_NAME	SERVICE_NAME	text(字符串)	数据长度:256	STD_MEDIA_SERVER_THRIFT_SERVICE	编辑 删除
属性	MANAGER_SERVICE_NAME	MANAGER_SERVICE_NAME	text(字符串)	数据长度:256	BVCRdpManagerThriftService	编辑 删除
属性	MEDIA_THRIFT_SERVICE	MEDIA_THRIFT_SERVICE	text(字符串)	数据长度:256	BVCRdpMediaThriftService	编辑 删除

服务列表

管理网关下产品的产品Profile服务调用时请求服务的目标地址。支持HTTP服务请求、WebSocket服务、RedisMQ服务。

添加服务

点击【添加服务】按钮，输入服务名称、目标服务的地址便可新增一个服务地址，建议此处输入地址为地址的通用部分，在产品Profile服务中使用服务地址时，再输入详细的接口名称。

HTTP类型服务地址和请求头支持**动态参数**定义，输入格式为\${参数名}（例如：/\${admin}）。若配置了动态参数，请在使用该服务地址的产品服务定义下“输入参数”列表中定义同名的参数（如admin）。实际请求时，系统将自动使用该参数的值替换地址中的动态部分，生成完整的请求URL。**示例说明：**接口地址：[http://api.example.com/\\${admin}/status](http://api.example.com/${admin}/status) 输入参数中定义：admin（值为visitor） 最终请求URL：<http://api.example.com/visitor/status>

实体管理 / 网关 / BVMedia

← BVMedia



服务高级配置（更多）

点击服务编辑/添加窗口中的 **更多** 折叠面板，可以对 HTTP 类型的服务进行更精细的请求报文配置与安全鉴权脚本编写：

- 1. **Params (Query 参数)**
- 2. 用于添加 URL Query 参数。点击 **+ 新增参数**，输入 Key-Value 对。这些参数将在实际请求时以 `?key=value` 的形式自动追加到服务地址末尾。
- 3. **Body (请求体)**
- 4. 用于配置 POST/PUT 等请求的请求体数据。支持以下四种格式：
 - **none**：不发送请求体（默认）。
 - **form-data**：以键值对表单数据形式发送（通常用于包含文件上传或标准表单提交）。
 - **x-www-form-urlencoded**：以 URL 编码的键值对形式提交请求体。
 - **json**：以标准 JSON 格式提交请求体。
- 5. **Headers (请求头)**
- 6. 用于添加自定义 HTTP 请求头。点击 **+ 新增参数**，输入请求头名称和对应的值（例如：配置 `X-Tenant-Id: tenant001`）。
- 7. **请求脚本 (Request Script)**
- 8. 支持使用 JavaScript (ECMAScript 6) 编写请求预处理脚本。在平台真正发起 HTTP 请求前，系统会调用该脚本以动态调整请求参数或进行加密签名。
- 9. **内置注入对象**：
 - **reqConf**：即将发送的 HTTP 请求配置对象（包含 `method`、`url`、`params`、`headers`、`body`、`contentType` 等属性，脚本中可直接对其进行读写修改）。
 - **Auth**：平台内置的鉴权工具类，提供常用的安全认证快捷方法（如 `Auth.bearer(reqConf, token)`、`Auth.basic(reqConf, username, password)`、`Auth.digest(reqConf, username, password)` 等）。
- 10. 更多详细的使用示例、Python 支持说明与内置 API 函数请参考专文：[请求脚本中使用Auth认证](#)。

WebSocket服务仅限使用系统内置“DTS内置服务”，您可以通过列表查看到连接的ip和port，完整地址为“ws://ip:port/{productid}/{deviceid}”。`productid` `deviceid` 为对应产品、实体的id。

发现实体接口列表

发现实体接口是可以通过Http接口方式请求获取第三方服务的实体(如物联网实体)的资源数据，经过脚本解析后成为DTS的实体。可以快速更新/获取实体列表。适用于设备不是第三方主动上报到DTS，而是通过接入网关主动向第三方请求。如接入海康视频监控平台，就可以通过此种方式主动的获得海康视频监控平台中的设备资源。

添加接口

点击 **【添加接口】** 按钮，在弹出的“编辑/创建接口”窗口中配置接口的基本信息、请求报文以及用于将数据转换为 DTS 格式的响应消息解析脚本：

- 1. **接口基本信息与请求配置**
- 2. **请求地址**：输入第三方服务的 HTTP 接口地址，选择请求方法（如 `GET`、`POST` 等）。
- 3. **请求参数配置**（包含 Params、Body、Headers、请求脚本四个子 Tab）：

- **Params (Query 参数)**: 配置 URL Query 参数 (如 `?limit=100`)，将自动拼接到接口请求地址尾部。
- **Body (请求体)**: 支持为 `POST` / `PUT` 请求配置 `json` (例如配置包含 `timeout` 和 `serviceTags` 的 JSON 报文)、`form-data` 等类型的请求体数据。
- **Headers (请求头)**: 配置请求的 Header 信息 (如 `Content-Type: application/json; charset=UTF-8`，`Accept: application/json`)。
- **请求脚本**: 支持使用 JavaScript (ECMAScript 6) 编写请求预处理脚本，在请求发送前动态修改请求配置对象 `reqConf` (具体内置对象与方法请参考服务列表中的[请求脚本说明](#))。

4. 响应消息解析脚本 (Response Script)

5. 用于接收并解析第三方服务返回的原始响应，将其映射并重构为符合 DTS 规范的实体数据集合 (JSON 数组)。

6. 函数定义: ``javascript /**

- 从网关发现实体，将第三方服务的自定义数据格式转换为DTS平台的BV JSON数据格式，解析后返回实体集合。
- `@param {Object} body` - 入参，接口返回的响应体对象 (系统已自动将其从 JSON 字符串反序列化为 JS 对象)
- `@returns {Array}` 出参，符合 DTS 规范的标准实体/设备 JSON 数组 */ function rawDataToProtocol(body) { // ... 过滤、循环转换与格式映射逻辑 ... return all_devices; } ``

7. 出参实体字段规范 (JSON 数组中的每个对象结构)：

- `deviceId` (String, 必传): 实体/设备编号，需保证平台内全局唯一 (如设备 UUID、物理出厂 MAC 或 GUID)。
- `deviceName` (String, 必传): 实体/设备显示名称，最大长度限制为 255。
- `deviceStatus` (Number, 非必传): 实体状态，`0` =禁用，`1` =启用 (默认)。
- `location` (String, 非必传): 实体位置 (默认 `"0 0 0"`)，格式为 `"纬度 经度 高程"`。
- `payload` (Object, 非必传): 实体的初始属性数据。放入这里的属性标识符 (如 `serialNumber`，`deviceStates` 等) 必须与该产品 Profile 中定义的属性标识符一致，否则无法成功记录。

响应消息解析脚本编写示例

以下为一个通过 `POST` 请求向第三方“看门狗视频管理器”拉取 VSS 视频设备列表，并将其解析为 DTS 标准视频实体集合的脚本示例：

```
function rawDataToProtocol(body) {
  // 打印第三方接口返回的原始数据以便调试
  console.log(body);

  var all_devices = []; // 用于收集解析转换后的标准设备列表

  try {
    var resultObj = body;

    // 核心步骤 1: 判断并提取第三方返回结构中的设备集合
    // 本示例中，第三方设备的返回路径为: ReturnValue.mediaDeviceList
    if (resultObj && resultObj.ReturnValue && resultObj.ReturnValue.mediaDeviceList) {
      var deviceList = resultObj.ReturnValue.mediaDeviceList;

      // 核心步骤 2: 遍历设备列表，将第三方属性映射到 DTS 格式
      for (var k = 0; k < deviceList.length; k++) {
        var device = deviceList[k];

        var formatted_device = {
          "deviceId": device.deviceGuid || "", // 设备唯一 GUID
          "deviceName": device.deviceName || "", // 设备名称
          "deviceStatus": (device.deviceStates !== undefined ? device.deviceStates : -1), // 设备状态
          "payload": {
            // payload 内部属性键名需与产品 Profile 属性严格对应
            "factoryGuid": device.factoryGuid,
            "deviceModel": device.deviceModel,
            "serialNumber": device.serialNumber,
            "deviceStates": device.deviceStates,
            "mediaSourceGuid": "E8565B1B-8762-E84D-805A-4C515183E829",
            "spacePosition": device.spacePosition,
            "deviceType": device.deviceType
          }
        };
        all_devices.push(formatted_device);
      }
    }
  }
}
```

```
    all_devices.push(formatted_device);
  }
} catch (e) {
  console.error("解析响应脚本时出错: " + e.message);
}

// 核心步骤 3: 返回重构完毕的标准设备列表
return all_devices;
}
```

配置完毕并编写解析脚本后，点击 **【保存】**。您可以在接口列表中点击该接口的调试/测试按钮，模拟请求与响应解析，以确保解析脚本无运行时错误。

网关信息 网关属性 服务列表 发现实体接口

+ 添加接口

请输入接口名称

接口名称	请求类型	接口地址	描述	创建时间	操作
未命名接口444	POST	http://192.168.200.237:4523/m1/47...	1212123434343434	2025-01-14 19:31:57	编辑 删除
获取设备列表Post	POST	http://192.168.100.236:4523/m1/47...	通过apifox测试设备添加post请求1212	2024-11-25 17:28:11	编辑 删除
获取设备列表Get	GET	http://192.168.100.236:4523/m1/47...	通过apifox获取设备列表	2024-11-25 17:18:22	编辑 删除

请求脚本中使用Auth认证

请求脚本中 Auth 认证使用指南

适用范围

`Auth` 只在请求脚本中可用。

请求脚本用于在平台真正发起第三方 HTTP 请求前，修改当前请求配置 `reqConf` 。它适用于网关服务、网关环境等带有 `requestScript` / `requestScriptMethod` 的 HTTP 请求链路。

内置对象

请求脚本运行时会自动注入两个对象：

`reqConf`
`Auth`

`reqConf` 是即将发送的 HTTP 请求配置，结构如下：

```
{
  "method": "POST",
  "url": "https://example.com/api",
  "params": {},
  "headers": {},
  "body": {},
  "contentType": 1
}
```

字段说明：

字段	说明
<code>method</code>	HTTP 方法，如 <code>GET</code> 、 <code>POST</code> 、 <code>PUT</code> 、 <code>DELETE</code>
<code>url</code>	请求地址
<code>params</code>	URL query 参数
<code>headers</code>	请求头
<code>body</code>	请求体
<code>contentType</code>	请求体类型： <code>0</code> form， <code>1</code> json， <code>2</code> multipart， <code>3</code> none

请求脚本不需要 `return` ，直接修改 `reqConf` 即可。脚本执行结束后，平台会把修改后的 `reqConf` 回写并用于实际 HTTP 请求。

Auth 方法

Bearer Token

JS:

```
Auth.bearer(reqConf, "your-token");
```

Python:

```
Auth.bearer(reqConf, "your-token")
```

最终效果:

```
Authorization: Bearer your-token
```

Basic Auth

JS:

```
Auth.basic(reqConf, "username", "password");
```

Python:

```
Auth.basic(reqConf, "username", "password")
```

最终效果:

```
Authorization: Basic base64(username:password)
```

API Key 请求头

JS:

```
Auth.apiKeyHeader(reqConf, "X-API-Key", "your-api-key");
```

Python:

```
Auth.apiKeyHeader(reqConf, "X-API-Key", "your-api-key")
```

最终效果:

```
X-API-Key: your-api-key
```

API Key Query 参数

请求脚本中使用Auth认证

JS：

```
Auth.apiKeyQuery(reqConf, "access_token", "your-token");
```

Python：

```
Auth.apiKeyQuery(reqConf, "access_token", "your-token")
```

最终请求 URL 会追加：

```
?access_token=your-token
```

Digest Auth

JS：

```
Auth.digest(reqConf, "username", "password");
```

Python：

```
Auth.digest(reqConf, "username", "password")
```

Digest 会先基于当前 `reqConf` 发起一次未认证请求，用响应头中的 `WWW-Authenticate` 计算摘要，再写入：

```
Authorization: Digest ...
```

注意：

- `reqConf.method` 和 `reqConf.url` 必须有效。
- 目标服务必须返回 Digest challenge，也就是 `WWW-Authenticate: Digest ...`。
- `auth-int` 模式不支持 `multipart/form-data` 请求体。

无认证

JS：

```
Auth.none(reqConf);
```

Python：

```
Auth.none(reqConf)
```

`Auth.none` 不会主动删除已有认证信息，只会保持请求配置结构可用。

常见脚本示例

JS：Bearer Token 并追加租户头

请求脚本中使用Auth认证

```
Auth.bearer(reqConf, "your-token");

reqConf.headers["X-Tenant-Id"] = "tenant001";
reqConf.params["page"] = 1;
reqConf.body["deviceId"] = "demo-device";
```

JS：接口要求 API Key 放 Query

```
Auth.apiKeyQuery(reqConf, "api_key", "your-api-key");

reqConf.params["timestamp"] = Date.now();
```

JS：完全手写 Authorization

不使用 `Auth` 也可以直接操作请求头：

```
reqConf.headers["Authorization"] = "Bearer " + "your-token";
reqConf.headers["X-Trace-Id"] = "trace-001";
```

Python：Basic Auth 并修改 JSON 请求体

```
Auth.basic(reqConf, "username", "password")

reqConf["headers"]["X-Tenant-Id"] = "tenant001"
reqConf["body"]["deviceId"] = "demo-device"
```

Python：无请求体 GET

```
Auth.bearer(reqConf, "your-token")

reqConf["method"] = "GET"
reqConf["contentType"] = 3
reqConf["body"] = None
reqConf["params"]["deviceId"] = "demo-device"
```

生效链路

请求脚本的处理顺序：

1. 平台先根据接口配置构造 `RequestConfig`。
2. 如果配置了 `requestScript`，调用 `executeRequestContent`。
3. 脚本修改 `reqConf`，可通过 `Auth` 写入认证信息。
4. 平台将脚本结果回写到 `RequestConfig`。
5. HTTP 执行器读取 `url`、`params`、`headers`、`body`、`contentType` 发起请求。

因此 `Auth` 的本质是修改 `reqConf.headers` 或 `reqConf.params`，不是独立的认证通道。

常见问题

1. 脚本里提示 `Auth is not defined`

常见原因：

- 当前脚本不是请求脚本，而是数据解析函数脚本。
- 当前链路没有走 `executeRequestContent` 。
- 环境配置 `script.bridge.allow-list` 限制了桥接对象，未允许 `Auth` 。

2. `Auth.bearer` 执行了但第三方仍返回 401

建议检查：

- token 是否带了多余的 `Bearer` 前缀；`Auth.bearer` 会自动拼接。
- 第三方是否要求其他 header，例如租户、应用编号、时间戳、签名。
- 当前接口是否实际走 HTTP 请求链路，而不是 Redis、WebSocket、MCP 等其他链路。
- 请求脚本后续代码是否覆盖了 `reqConf.headers["Authorization"]` 。

3. Digest 认证失败

建议检查：

- 目标服务第一次响应是否包含 `WWW-Authenticate: Digest ...` 。
- `reqConf.url` 是否为完整可访问地址。
- `reqConf.method` 是否正确。
- 目标服务是否使用当前实现支持的算法：`MD5`、`MD5-sess`、`SHA-256`、`SHA-256-sess` 。

4. 应该用 Auth 还是手写 header

常见认证建议优先使用 `Auth` ：

- Basic
- Bearer
- API Key Header
- API Key Query
- Digest

特殊签名算法、动态 HMAC、时间戳签名等场景，可以在请求脚本中自行计算并写入：

```
reqConf.headers["Authorization"] = "Custom " + signature;
reqConf.headers["X-Timestamp"] = String(Date.now());
```

快速选择

第三方认证方式	推荐写法
Bearer Token	<code>Auth.bearer(reqConf, token)</code>

第三方认证方式 Basic Auth	推荐写法 Auth.basic(reqConf, username, password)
请求头 API Key	Auth.apiKeyHeader(reqConf, key, value)
URL 参数 API Key	Auth.apiKeyQuery(reqConf, key, value)
Digest Auth	Auth.digest(reqConf, username, password)
自定义签名	手动写入 reqConf.headers 或 reqConf.params

实体

实体

产品指某一类实体，创建完产品后，需要为实体创建身份。可以创建单个实体，也可以批量创建实体。这里是手动为产品下添加实体，获取实体证书信息用于后续的数据交换。也可以通过实体接入主动上报实体到DTS。

数字孪生服务 DTService

首页

实体管理

产品

网关

实体

组合实体

低代码设计器

分组管理

组织结构

任务管理

订单管理

系统管理

实体管理 / 实体

实体

产品: 请选择产品 网关: 请选择网关 实体总数 755

添加实体

批量添加

DeviceID

请输入DeviceID查询

请选择标签

☐	实体名称	DeviceID	所属产品	启用状态	创建时间	最近更新时间	操作
☐	全景	022C951EB834609D0AA03D8F2FDD0230	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	钻石广场-主	7D5C17F387D69C0F9C2892E67BF2D424	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	东北楼	266AE090-2F48-8345-AF3E-0F2D20D07C09	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	大门入口球机	7061396DF993C94960DB83842D39DDF3	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	三洲楼大沙盘全景相机	104383D5AB094F110766A4731AFD984E	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	演播	3A724D55-E272-CS4C-8789-BC8661ADF017	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	室内相机07(广角)	6B948EFCABCF519C59C12CDD5588A956	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	室内相机08(人脸识别)	E9C9939C5A306175831FFC63DD40CA03	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	室内相机09	14B20D3B49EA48B13B092A8F00370C79	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除
☐	钻石广场-辅	E68FC0D9CBE48DF6057CA169B95FE7E	bv相机2	已启用	2025-03-27 09:51:54	2025-03-27 09:51:54	编辑 删除

☐ 删除 ☐ 禁用 ☐ 启用

共755条, 每页显示 10 << 1 >> GO

单个创建实体

添加实体

特别说明: DeviceID可以为空, 当为空时, 数字孪生模块会颁发全局唯一标识符作为DeviceID, 网关可以不选, 后期可在产品实体列表进行绑定。

* 产品

请选择产品

网关

请选择网关

DeviceID

自动生成

* 实体名称

取消

确定

EFCABCF519C59C12CDD55B8A956bv相机2

参数	描述
产品	选择产品。新创建的实体将继承该产品定义好的功能和特性。
网关	选择网关。新创建的实体将继承该网关定义好的公共参数（属性）。
deviceId	设置实体名称。实体名称在产品内具有唯一性。实体名称长度 为4-32个字符，可包含英文字母、数字和特殊字符，包括连接号（-）、下划线（_）、at符号（@）、点号（.）和英文冒号（:）。 说明 deviceId可以为空。为空时，由DTS生成一个全局唯一标识符作为实体的deviceId。
备注名称	设置备注名称。备注名称长度为4-64个字符，可包含中文汉字、英文字母、数字和下划线（_）。一个中文汉字算2字符。

实体创建成功后，可以查看、复制实体证书信息。实体证书由实体的productId、deviceId和deviceSecret组成，是实体与DTS进行通信的重要身份认证，建议妥善保管。

参数	说明
productId	实体所隶属产品的Key，即DTS为产品颁发的全局唯一标识符。
deviceId	实体在产品下的唯一标识符。deviceId与实体所属产品的productId组合，作为实体标识，用来与DTS进行连接认证和通信。
deviceSecret	DTS为实体颁发的实体密钥，用于鉴权认证。需与deviceId成对使用。

批量创建实体

批量添加实体。

批量添加设备

×

实体

* 产品

请选择产品

* 功能类型

☒ 自动生成

☐ 批量上传

☐ 从网关发现设备

* 设备数量

请输入设备数量

取消

确定

自动生成：无需为实体指定名称。填写实体数量后，系统将为每个实体自动生成由字母、数字随机组合成的deviceId。

批量上传：需要为所有实体指定名称。单击下载.csv模板下载表格模板，在模板中填写实体名称，然后将填好的表格上传至控制台。

从网关发现实体：选择创建的网关，设定需要请求的接口地址，系统将自动请求网关下的实体。根据选择的产品profile自动生成实体。

批量添加设备

×

* 产品

请选择产品

▼

* 功能类型

☐ 自动生成

☐ 批量上传

☒ 从网关发现设备

网关

智能冰箱

▼

请输入设备数量

请求接口

请选择请求接口

▼



说明 填写模板时，请注意： 1、实体名称长度为4-32个字符，可包含英文字母、数字和特殊字符，包括连接号（-）、下划线（_）、at符号（@）、点号（.）和英文冒号（:）。 2、实体名称在产品维度具有唯一性。列表中的实体名称不可重复，且不可与该产品下已有实体的名称重复。 3、一个文件中最多可包含1,000个名称。 4、文件大小不超过2 MB。

若批量上传的实体名称列表中有不合法的名称，将出现错误提示。请单击**下载不合法列表**，查看不合法的实体名称。根据实体名称规范，修改实体名称，再重新上传文件。

编辑实体

在实体页面选择需要编辑查看的实体，点击“编辑”进入实体详情页面。在实体详情页可查看到实体基本信息等内容。

产品 [bv相机2](#) [复制](#) ProductID [uiGWx5gCHlaa0oh8](#) [复制](#) DeviceSecret ***** [查看](#)

实体信息

Topic类列表

产品Profile

服务端订阅

实体信息 [编辑](#)

产品名称	bv相机2	ProductID	uiGWx5gCHlaa0oh8	创建时间	2025-03-27 09:51:54
DeviceID	022C951E8834609D0AA03D8F2FDD0230			更新时间	2025-03-27 09:51:54
实体名称	全景	IP地址			
当前状态	未激活	认证方式			

实体信息

查看实体的基本信息包含产品信息和网关信息以及授权信息。

实体管理 / 实体 / 室外采集器01

← 室外采集器01

产品 [BV相机](#) [复制](#) ProductID [WmdLBvoeh5H1IOpe](#) [复制](#) DeviceSecret ***** [查看](#)

实体信息

Topic类列表

产品Profile

服务端订阅

实体信息 [编辑](#)

产品名称	BV相机	ProductID	WmdLBvoeh5H1IOpe	创建时间	2025-03-27 09:51:53
DeviceID	1B76EE26DB7AB6DD866E6E27D265E712			更新时间	2025-03-27 09:51:54
实体名称	室外采集器01	IP地址			
当前状态	未激活	认证方式			

网关信息

网关	BVMedia	gatewayID	76441b8484664921	网关标识符	BVMedia
网关类型	BVMedia				

授权信息

被授权用户组		被授权用户			
--------	--	-------	--	--	--

标签 [编辑](#)

标签: [coordinate0 0 0](#)

实体标签

通常根据实体的特性为实体添加的特有标记。实体标签信息会跟随实体在系统内部流转。该功能支持“可选择 + 可输入”的组合模式，支持两级标签管理。

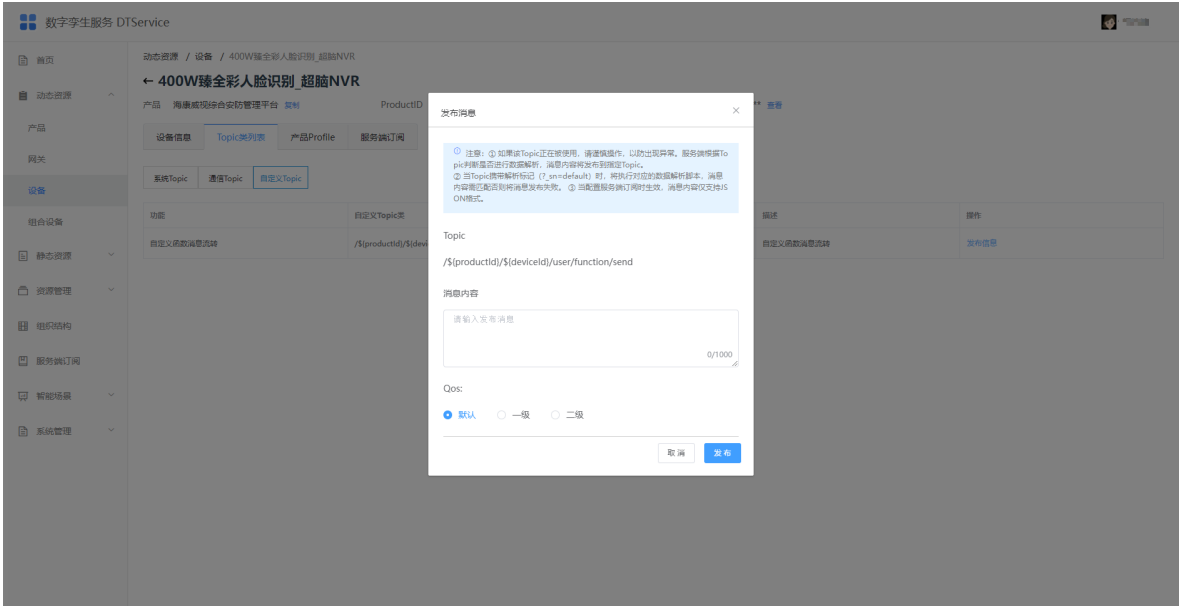
参数	说明
----	----

实体

参数	说明
标签分组 (一级标签)	标签分组名称，支持快速选择已有分组或自定义输入创建（如 Status 、 Vendor ）。
标签 (二级标签)	具体标签值，支持快速选择已有标签或自定义输入创建（如 Active 、 Hikvision ）。

Topic类

产品的所有Topic类会自动映射到实体上，无需单独为每个实体创建Topic。实体的自定义Topic可使用【发布信息】进行Topic消息的发送测试。



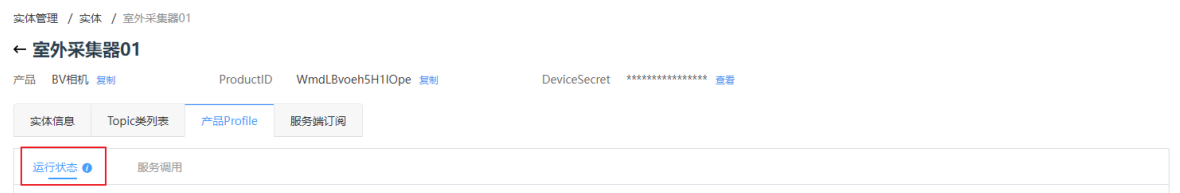
- ① 如果该Topic正在被使用，请谨慎操作，以防出现异常。服务端根据Topic判断是否进行数据解析，消息内容将发布到指定Topic。
- ② 当Topic携带解析标记（?_sn=default）时，将执行对应的数据解析脚本，消息内容需匹配否则将消息发布失败。
- ③ 当配置订阅关系时生效，消息内容仅支持JSON格式。

产品Profile

实体的产品Profile不可编辑和修改。所有实体的产品profile都与产品一致。

运行状态

查看实体属性的最新值，可设置实时刷新。



实体

<div>请输入属性名称</div>					实时刷新	
功能类型	属性名称	属性标识符	数据类型	数据定义	最新值	最近更新
属性	当前设备截图地址 <div>可选</div>	deviceImage	text(字符串)		http://172.27.240.1:44040/fms ge/20250325/c15641995.jpg	2025-03-24 17:00:07
属性	服务请求编号 <div>可选</div>	requestId	text(字符串)		d0f887851fb94652bb63925757 175a08	2025-03-13 18:00:08
属性	扩展数据 <div>自定义</div>	extendData	text(字符串)		["NetworkState": "1"]	2025-03-27 09:50:24
属性	设备编号 <div>自定义</div>	deviceGuid	text(字符串)			
属性	设备类型 <div>自定义</div>	deviceType	enum(枚举型)		0	2025-03-27 09:50:24
属性	设备名称 <div>自定义</div>	deviceName	text(字符串)			

服务调用

已在产品Profile中定义的服务，便可在实体下通过【发送请求】调用服务。需要设置服务调用的参数值以及回调的服务地址。此服务调用为一次性调用，常用于服务测试使用。

[illegible]

订阅关系

当前实体存在订阅时，可查看到所有订阅的信息。

实体管理 / 实体 / 全景

← 全景

产品

bv相机2

复制

ProductID

uiGWx5gCHlaa0oh8

复制

DeviceSecret

查看

实体信息

Topic类列表

产品Profile

服务端订阅

订阅名称	订阅类型	订阅方式	消息质量	创建时间	操作
暂无数据					

实体反控

实体反控

通过网关，DTS可以实现外部数据源到实体的状态和事件映射，在某些情况下，数字孪生还需要反向控制实体，这就需要实体将其开放的控制指令或处理逻辑以服务的方式提供，让用户不需要在应用层编程即可配置在任务中调用。通过本节可以了解到DTS对实体反向控制的流程。

支持多种方式反控实体，HTTP、WebSocket、Redis等方式。HTTP协议方式适合于目标服务与DTS服务网络能互相访问的情况；WebSocket方式适合于目标服务能访问到DTS，但DTS不能访问到目标服务的情况下。Redis方式适合于高并发、高实时、且服务双方均能连接到同一个 Redis 实例的分布式架构中。

基于HTTP方式进行实体反控

示例

本文以温湿度传感器产品为例，介绍使用 HTTP 协议服务，通过 DTS 进行实体反控的配置与调试流程。

[!IMPORTANT] 生产架构与调试说明 * 在生产环境/实际使用时：DTS 网关中配置的服务地址（URL）应当指向您**真实的物理设备、本地边缘网关，或者是您企业内部的实体控制系统服务接口**。* 在本文教程中：为了方便您在**没有真实物理硬件**的情况下能闭环体验并测试 DTS 的反控配置链路，我们采用 **Postman 或 APIFox 搭建本地 Mock 服务**来模拟设备端接收控制指令。这仅作为本地联调和报文格式检查的手段，实际生产开发时请替换为您的真实设备接口。

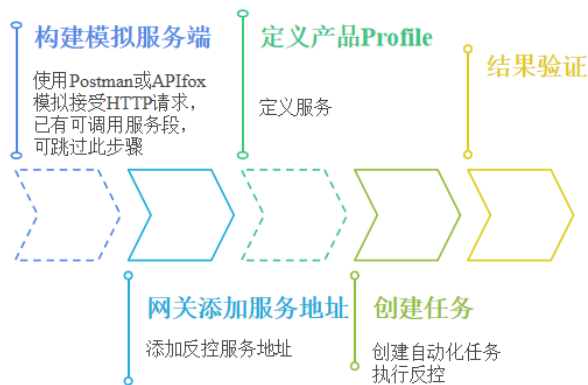
生产环境与调试环境配置对照：

配置项 / 概念	本文调试阶段（以 Postman 为例）	实际生产阶段（以智能插座重启为例）
反控服务承载者	Postman Mock Server (或 APIFox Mock)	物理设备本身（如设备内置 WebServer）或本地边缘网关 / 控制服务
网关服务地址 (URL)	<code>https://xxxx.mock.pstmn.io/reset</code> (Mock 服务提供的公网域名/接口)	<code>http://192.168.1.100:8080/api/reset</code> (物理设备在局域网/专网的真实IP与接口)
数据通信方向	DTS 平台 $\rightarrow$ 互联网 $\rightarrow$ Postman 云 Mock 接口	DTS 平台 $\rightarrow$ 局域网/专网 $\rightarrow$ 真实物理设备接口

反控配置流程包含以下几个主要步骤：

- i. 准备反控接收服务端：生产环境中为您的物理设备或网关反控接口，本文调试时使用 Postman/APIFox 模拟；
- i. 网关添加服务地址：在通用网关的服务列表中添加上述反控服务地址；
- i. 定义产品Profile：在产品中定义支持的服务（如重启复位服务）；
- i. 创建任务：创建自动化任务以触发并执行服务调用动作；
- i. 结果验证：手动或自动执行任务，验证反控结果。

基于HTTP方式的实体反控流程



准备事项

- 1.注册数字孪生服务DTService（以下简称DTS）账户
 - 2.准备Postman或APIFox，用于模拟接收设备反向控制的指令。
 - 3.已成功创建需要进行反控的产品“温湿度传感器”。请参考[实体接入](#)
 - 4.已成功创建需要进行反控的设备。

构建模拟服务端（调试步骤，若已有真实设备接口可跳过）

已有可调用的服务端，可忽略此步骤。使用Postman模拟可以查看到DTS请求服务时，发送的数据内容。如不需要查看，可以使用APIFox模拟。

Postman模拟服务端

1. 使用Postman模拟一个接收实体反向控制的服务端“重启复位服务”。模拟接收某个实体发送的重启复位指令。
2. 在右侧菜单栏中选择“Collections”，点击新建一个集合，命名为“模拟服务端”。
3. 在“模拟服务端”集合下，新建一个Post方式的HTTP接口，命名为“重启复位服务”。
4. 定义接口地址为“/reset”，请求参数有 `deviceid` ,返回内容为 `code` `message` `info` 。
5. 定义完成后，保存接口。

6. 在右侧菜单栏中选择“Mock servers”，新建一个Mock server，命名为“MyMockServer”，选择“模拟服务端”集合作为

Create a mock server

Select a collection to mock

☒ An existing collection ☐ Create a new collection

Mock server name

MyMockSevrer

Collection

模拟服务端

Environment

An environment is a group of variables useful for storing and reusing values.

No Environment

Simulate a fixed network delay

No delay

☐ Save the mock server URL as an new environment variable

This will create a new environment containing URL.

☐ Make mock server private

To call a private mock server, you'll need to add an `x-api-key` header to your requests. See how to generate a Postman API key ↗

Create Mock Server

Cancel

MockServer下的集合。

7. 新建完成后，点击“Copy URL”，复制Mock server的地址。
8. 将复制的mockserver地址，补充到接口“重启复位服务”中，作为接口的请求地址。示例如，“https://3cb98ad9-2243-4c27-b0c1-c5bb0f8af23f.mock.pstmn.io/reset”。
9. 在右侧侧边栏中选择“智能插座”，点击“更多”添加一个示例，命名为“reset”。
10. 回到“Mock servers”下，进入“MyMockServer”，等待DTS的服务请求。

APIfox模拟服务端

使用APIfox模拟一个接收实体反向控制的服务端“重启复位服务”。模拟接收某个实体发送的重启复位指令。

1. 点击新建一个接口，命名为“重启复位服务”。
2. 定义接口地址为“/reset”，请求参数有 `deviceid`，返回参数为 `code` `message` `info`。
3. 将接口采用“本地Mock环境”或“云Mock环境”启动，保证能被DTS所访问。
4. 查找到接口的Mock地址，复制已备用。

示例采用本地Mock环境，实际使用时，将需修改为具体的IP地址。保证目标地址与DTS之间网络通信正常。

查询功率服务

POST

/queryPowerService

开发中

共享

创建时间 2025年4月28日 修改时间 几秒前 修改者 ytt001 创建者 ytt001 责任人 未设置 目录 智能插座通用网关接入Demo

Mock

本地 云端 自托管

名称	来源	URL / 参数
成功 (200)	接口响应	http://127.0.0.1:4523/m1/6293501-5988289-default/queryPowerService

请求参数

Body 参数 application/json	示例
<div>deviceid string 必需</div>	<pre>{ "deviceid": "string"}</pre>

返回响应

成功(200)

HTTP 状态码: 200 内容格式: JSON application/json	示例
<div>数据结构</div> <div>code integer 必需</div> <div>message string 必需</div> <div>info string 必需</div>	<pre>{ "code": 0, "message": "success", "info": "ok"}</pre>

网关添加服务地址

1. 进入实体管理模块，选择网关。

2. 选择“通用网关”，点击编辑进入通用网关的详情页面->选择服务列表 Tab页面。

3. 在服务列表 Tab页面下，选择添加一个服务地址。

4. 命名为“本地Postman服务”，选择类型为Http方式，服务地址输入已通过Postman创建好的服务“MyMockServer”的Mock地址。若使用APIfox方法一致，输入APIFox提供的Mock地址即可。

这个服务地址需指向接收控制服务请求的目标地址。

数字孪生服务 DTService

实体管理 / 网关 / BV通用网关

← BV通用网关

GatewayKey 1ca63711958e430c 复制 GatewaySecret ***** 重置 实连接 89 前往管理

网关信息 网关属性 服务列表 发现本地接口

添加服务

服务名称	类型	服务地址	操作
测试删除	HTTP	http://192.168.100.37:9090	编辑 删除
本地Mock服务	HTTP	http://192.168.100.236:4523/m1/6293501-5988289-default	编辑 删除
远程Postman服务	HTTP	https://3d99fa29-2243-4c27-b0c1-c3ba09fa23f5-mock.postman.io	编辑 删除
Ofema	HTTP	http://192.168.100.82:11434	编辑 删除
AICenterServer	HTTP	http://192.168.100.37:5004	编辑 删除
智慧云指挥	HTTP	https://dashscope.aliyuncs.com	编辑 删除
DTS内置服务	Websocket	wss://192.168.100.37:44050	编辑 删除
智能场景开关控制	HTTP	http://192.168.100.236:4523/m1/4795394-4449583-default	编辑 删除

定义产品Profile

需在产品Profile中定义实体支持的服务。服务执行后，根据使用场景选择结果返回方式：

- 支持通过属性上报或事件上报的方式，将执行结果反馈到DTS中。采用此方式，需定义对应的属性或事件。
- 支持通过HTTP或RedisMQ的方式，将执行结果转发给第三方平台；

在产品Profile所有编辑完成后，需点击**发布**。发布后的产品Profile才能被DTS识别。智能插座的数据格式为“BV JSON”，示例默认都使用BV JSON进行数据交互。无论数据格式是BV JSON或透传/自定义流程类似，BV JSON无需写消息解析脚本。

定义产品Profile（服务）

- 进入实体管理模块，选择产品。
- 选择“温湿度传感器”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
- 点击编辑草稿，选择添加功能，选择功能类型为**服务**。
- 输入功能名称为“重启复位”，设置标识符为resetDevice。
- 服务地址选择已定义的服务地址“本地Postman服务”，输入地址接收请求的接口地址“/reset”。
- 根据“重启复位服务”服务设置输入参数 deviceid。
- 根据需要设定选择结果返回方式为属性上报或事件上报。

* 功能类型

服务

* 功能名称

设备重启复位

* 标识符

基于HTTP方式进行实体反控

resetDevice

服务地址 ⓘ

本地Postman服务 https://3cb98ad9-2243-4c27-b0c1-c5bb0f8af23f.mock.pstl ▼

查看完整服务地址

接口

POST

/rest

调用方式

☐ 同步

☒ 异步

* 结果返回方式 ⓘ

属性上报 ▼

输入参数

参数名称: 设备编号

编辑 | 删除

+ 增加参数

输出参数

+ 增加参数

描述

请输入描述

0/100

图1：采用属性上报返回结果

* 功能类型

服务

* 功能名称

设备重启复位

* 标识符

resetDevice

基于HTTP方式进行实体反控

服务地址 ⓘ

本地Postman服务 https://3cb98ad9-2243-4c27-b0c1-c5bb0f8af23f.mock.pstl ▼

[查看完整服务地址](#)

接口

POST	/rest
------	-------

调用方式

☐ 同步 ☒ 异步

* 结果返回方式 ⓘ

事件上报 ▼

目标事件 ⓘ

获取重启状态 ▼

输入参数

参数名称: 设备编号	编辑 删除
------------	---

[+ 增加参数](#)

输出参数

[+ 增加参数](#)

描述

请输入描述

0/100

图2：采用事件上报返回结果

同步方式为立即返回结果，异步方式可有延迟。DTS会携带回调参数发送到服务。

方式1：结果返回方式为属性

定义产品Profile（属性）

1. 进入实体管理模块，选择产品。
2. 选择“智能插座”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为属性。

4. 输入功能名称为"重启状态"，设置标识符为 `status` 。

* 功能类型

属性

* 功能名称

重启状态

* 标识符

status

* 数据类型

bool(布尔型)

* 布尔值

false - false

true - true

* 读写类型

☐ 只读

☒ 读写

描述

用于接收重启复位服务的结果反馈

15/100

5. 确定保存属性。

方式2：结果返回方式为事件

定义产品Profile（事件）

1. 进入实体管理模块，选择产品。
2. 选择“智能插座”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为**事件**。
4. 输入功能名称为“获取重启状态”，设置标识符为getResult。
5. 设置输出参数 `status`。

* 功能类型

事件

* 功能名称

获取重启状态

* 标识符

getResult

* 事件类型

☒ 信息 ☐ 告警

* 消息质量

☒ 默认 ☐ 一级 ☐ 二级

输出参数

参数名称: 重启状态

编辑 | 删除

+ 增加参数

操作

☐ 持久化存储

描述

使用事件接收重启服务的反馈结果

15/100

6. 确定保存事件。

配置消息解析脚本

仅需为使用透传/自定义协议的产品实体配置。

消息解析脚本示例

```
/**
 * 将透传/自定义数据格式转换为DTS平台的BV JSON数据格式，实体注册、属性、事件上报到DTS时调用。
 * @param {JSONObject} body - 入参
 * @param {JSONObject} body._bv_callback_message - 服务回调参数（回调时携带）
 * @param {string} body._bv_callback_message.productId - 产品编号
 * @param {string} body._bv_callback_message.deviceId - 实体编号
 * @param {string} body._bv_callback_message.identifier - 事件的唯一标识符（回调方式为事件上报时携带）
 * @param {JSONObject} body._bv_callback_message._bv_service_callback - 服务调用参数
 * @param {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * @param {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识符
 * @param {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * @param {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * @param {number} flag - 入参类型标识，0=实体注册，1=属性上报，2=事件上报
 * @returns {JSONObject} 出参，BV JSON数据格式，不能为空
 */
function rawDataToProtocol(body, flag) {
    var inputData = body;
    var jsonObj = {};
    if (flag === 0) {
        /**
         * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
         * @returns {string} return.deviceName - 实体名称（最大长度255），必传
         * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
         * @returns {number} return.deviceType - 实体类型，0=动态资源，1=静态资源，必传
         * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
         * @returns {string} return.version - 版本号（默认default），非必传
         * @returns {JSONObject} return.payload - 扩展数据，用于更新实体属性值，数据基于产品 Profile 定义的属性过滤（已定义属性对应的数据保留，未定义属性对
         */
        jsonObj['deviceId'] = inputData.guid;
        jsonObj['deviceName'] = inputData.name;
        jsonObj['deviceStatus'] = inputData.status;
        jsonObj['payload'] = inputData;
        return jsonObj;
    }
    if (flag === 1) {
        /**
         * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
         * @returns {string} return.deviceSecret - 实体签名（不传时将系统匹配），非必传
         * @returns {string} return.deviceName - 实体名称（最大长度255），非必传
         * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
         * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
         * @returns {number} return.timestamp - 上行数据传输时间戳。格式为数值，值为从UTC 1970-01-01 00:00:00开始到当前时刻的毫秒数，非必传
         * @returns {string} return.version - 版本号（默认default），非必传
         * @returns {JSONObject} return.payload - 扩展数据，用于更新实体属性值，数据基于产品 Profile 定义的属性过滤（已定义属性对应的数据保留，未定义属性对
         */
        jsonObj['deviceId'] = inputData.guid;
        jsonObj['deviceStatus'] = inputData.status;
        jsonObj['payload'] = inputData;
    }
}
```

```
        return jsonObj;
    }
    if (flag === 2) {
        /**
         * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
         * @returns {string} return.deviceSecret - 实体签名（不传时将系统匹配），非必传
         * @returns {string} return.deviceName - 实体名称（最大长度255），非必传
         * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
         * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
         * @returns {string} return.identifier - 事件的唯一标识符，必传
         * @returns {number} return.timestamp - 上行数据传输时间戳。格式为数值，值为从UTC 1970-01-01 00:00:00开始到当前时刻的毫秒数，非必传
         * @returns {string} return.version - 版本号（默认default），非必传
         * @returns {JSONObject} return.payload - 扩展数据，事件的输出参数，数据基于产品 Profile 定义的事件输出参数过滤（已定义输出参数对应的数据保留，未定
        */
        jsonObj['deviceId'] = inputData.guid;
        jsonObj['identifier'] = inputData.notificationType;
        jsonObj['payload'] = inputData.content;
        jsonObj['timestamp'] = inputData.timestamp;
        return jsonObj;
    }
}
```

创建任务

1. 在任务管理页面，创建一个任务名为“温湿度定时重启”的场景。

任务逻辑

每个工作日晚上18点 触发 执行任务 重启复位服务

1. 创建完成后，进入编辑页面。根据任务逻辑编辑对应的规则。设定触发条件为每个工作日18点，添加执行任务类型为“服务调用”，选择温度传感器产品下的某一实体或全部实体，执行服务“重启复位服务”。

温湿度定时重启

保存

定时重启温湿度传感器

如果满足一下任一触发条件

定时触发

每天

🕒 18:00:00

选择器

且满足以下全部生效条件

就执行以下任务

1. 设备控制-服务调用

温湿度传感器 > 全部设备

执行服务

设备重启复位

参数配置

设置服务调用的参数

- 1. 设置服务调用的参数， `deviceid` 使用插入变量的方式，设置其值为温湿度传感器下的当前实体（执行此任务的实体）。

参数配置

配置服务调用时对应的输入参数的参数值,参数值可从多种来源中选择。请根据服务参数数据类型匹配正确的参数,系统将自动获取该参数值。

服务调用

设备重启复位

输入参数

选择参数值进行配置，建立映射

text(字符串)

设备编号

当前实体.产品编...

当前实体.产品编号

取消

确定

- 1. 保存任务，启动任务。手动执行任务，进行任务的检查。
- 2. 前往任务-执行日志，查看任务的执行情况。

Postman查看请求

在Postman中查看是否接收到来自DTS的请求。 `_bv_callback_message` 发送的回执参数。从中可获取到结果返回的方式 `_bv_callback_method` 以及结果返回地址 `_bv_callback_url`。以及执行服务所需的输入参数。详细参数解释可查看[回执参数](#)。

结果返回方式为属性上报

Time	Request	Response
Today at 4:01:55 PM	POST /reset	reset
Request headers		
Request body		
<pre>{_bv_callback_message: {...}, _bv_callback_method: "http", deviceID: "407ddc2c-1860-4afe-a26a-cfb06228016c"} _bv_callback_message: {...} _bv_service_callback: {...} identifier: "resetDevice" requestId: "0df8f52b51a54dab86f8912224472202" version: "default" timestamp: 1752048115881 productId: "MNGRcylFFmLxxvJh" method: 0 deviceId: "407ddc2c-1860-4afe-a26a-cfb06228016c" _bv_callback_method: "http" deviceId: "407ddc2c-1860-4afe-a26a-cfb06228016c"</pre>		



图1: BV JSON数据格式

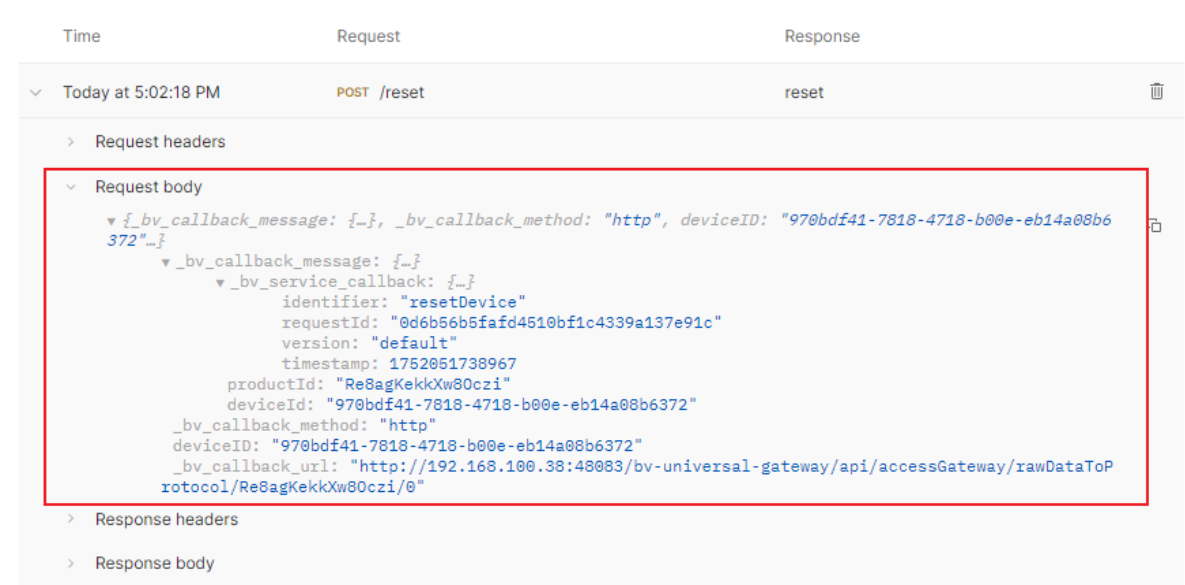


图2: 透传/自定义数据格式

结果返回方式为事件上报





图1: BV JSON数据格式

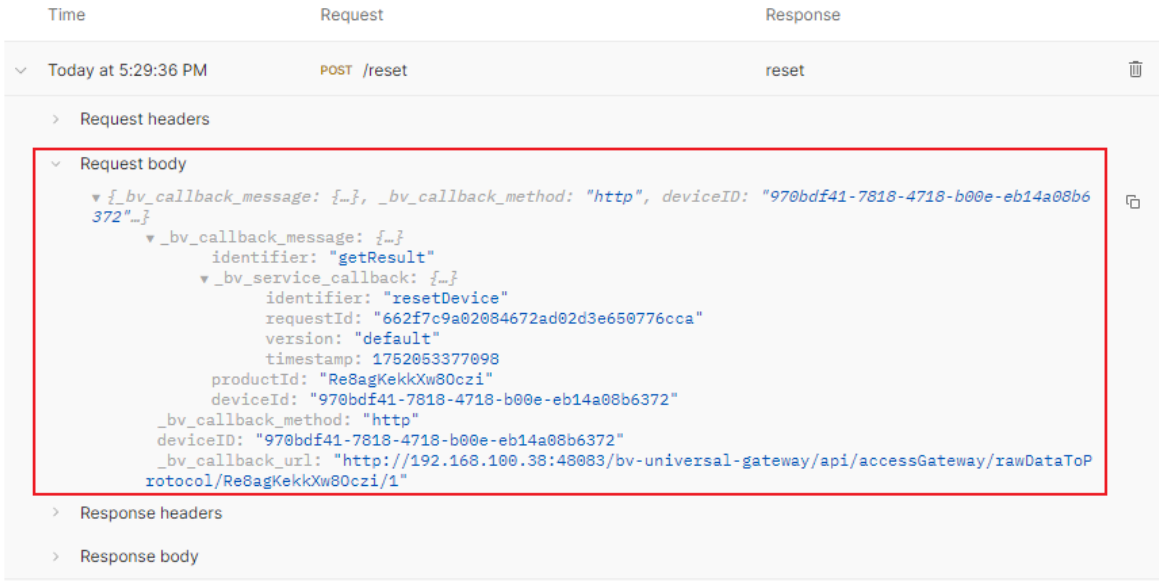


图2: 透传/自定义数据格式

若服务端并未收到来自DTS的服务请求，请检查服务定义以及目标服务与DTS之间的网络是否通畅。

APIFox模拟设备上报

- 根据HTTP协议的HTTP协议实体接入(BV JSON数据格式)-属性上报接口、HTTP协议实体接入(BV JSON数据格式)-事件上报接口，在APIFox中编写返回结果。
- 接口地址填写 `_bv_callback_url` 的值，请求体中 `productId` `identifier` `deviceId` 均来自 `_bv_callback_message`。
- 接口准备完毕，点击发送。

属性上报发送请求体示例

```
{
  "method": 0,
  "productId": "MNGRCylFFmLxxvJh",
  "deviceId": "407ddc2c-1860-4afe-a26a-cfb06228016c",
  "deviceStatus": 0,
  "payload": {
    "status": true
  }
}
```

事件上报发送请求体示例

```
{
```

```
{
  "method": 1,
  "productId": "MNGRCyIFFmLxxvJh",
  "deviceId": "407ddc2c-1860-4afe-a26a-cfb06228016c",
  "identifier": "getResult",
  "deviceStatus": 0,
  "payload": {
    "status": true
  }
}
```

若数据格式为透传/自定义，则需根据[HTTP协议实体接入\(透传/自定义\)-属性上报接口](#)、[HTTP协议实体接入\(透传/自定义\)-事件上报接口](#)，在APIFox中编写返回结果。

返回服务结果时，尽量携带回执参数。在透传/自定义数据格式下，可根据回执参数进行特殊的处理逻辑。

属性上报发送请求体示例

```
{
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "status": 0,
  "resetStatus": true,
  "_bv_callback_message": {
    "_bv_service_callback": {
      "identifier": "resetDevice",
      "requestId": "0d6b56b5fafd4510bf1c4339a137e91c",
      "version": "default",
      "timestamp": 1752051738967
    },
    "productId": "Re8agKekkXw8Oczl",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372"
  }
}
```

事件上报发送请求体示例

```
{
  "deviceType": "SmartRefrigerator",
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "notificationType": "getResult",
  "content": {
    "status": true
  },
  "timestamp": 1752053423202,
  "_bv_callback_message": {
    "identifier": "getResult",
    "_bv_service_callback": {
      "identifier": "resetDevice",
      "requestId": "662f7c9a02084672ad02d3e650776cca",
      "version": "default",
      "timestamp": 1752053377098
    },
    "productId": "Re8agKekkXw8Oczl",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372"
  }
}
```

结果验证

属性上报的结果可以在实体状态中直接查看，也可以经过[订阅管理](#)将向DTS订阅实体的属性变化和事件发生的数据，订阅端收到数据之后进行可视化处理便能对设备进行实时数据监控。DTS数据孪生架构提供快速接入查看DTS设备数据的平台功能[低代码设计器](#)（无需订阅），详细可视化功能可查看《[BigView低代码设计器教程](#)》

基于WebSocket方式进行实体反控

示例

本文以温湿度传感器产品为例，介绍使用 WebSocket 协议通道，通过 DTS 进行实体反控的配置与调试流程。

[!IMPORTANT] 生产架构与调试说明 * 在生产环境/实际使用时：建立 WebSocket 连接的客户端应当是运行在您**真实物理设备或边缘网关上的连接代码**（例如使用 Python/C++/Java 编写的 **WebSocket Client 进程**）。设备上电启动后需自动建立与 DTS 平台内置 WebSocket 服务端（由网关通信模块承载）的连接，以便持续监听并接收控制指令。* **在本文教程中**：为了方便您在**没有真实物理硬件**的情况下能闭环体验并测试 WebSocket 控制通道的配置链路，我们采用 **APIFox (或网页端 WebSocket 调试工具)** 作为**客户端模拟器**，来手动发起连接并查看 DTS 下发的控制消息。这仅作为本地联调和数据报文解析的手段，实际生产开发时请替换为您的真实设备客户端代码。

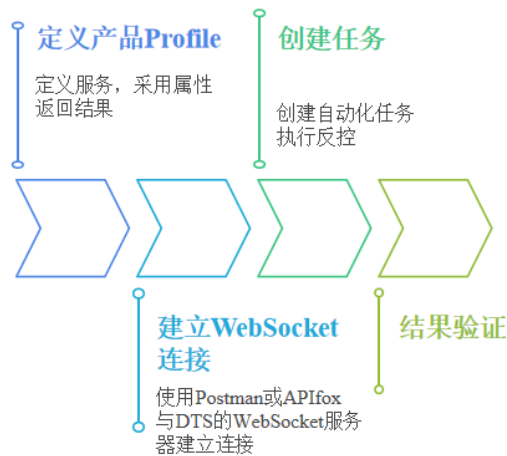
生产环境与调试环境配置对照：

配置项 / 概念	本文调试阶段（以 APIFox 为例）	实际生产阶段（以温湿度传感器客户端为例）
WebSocket 客户端	APIFox WebSocket 调试界面	运行在温湿度传感器设备（或网关）上的 Python/C++/Java 客户端程序
网络连接建立	读者在客户端中手动输入 URL 并点击“连接”	设备上电后，客户端程序自动发起并建立向 DTS 的 WebSocket 长连接
指令接收与处理	在 APIFox 的发送与接收日志面板中查看 JSON 报文	设备程序监听该长连接，解析收到的 JSON 报文并调用底层硬件接口执行物理操作

反控配置流程包含以下几个主要步骤：

- - i. 定义产品Profile：在产品中定义支持的服务（如重启复位服务，服务地址选择 DTS 内置服务地址）；
- - i. 建立WebSocket连接：在本地启动 WebSocket 客户端建立长连接（生产中为真实设备连接代码，本文调试时使用 APIFox 模拟）；
- - i. 创建任务：创建自动化任务以触发并执行服务调用动作；
- - i. 结果验证：手动或自动执行任务，在客户端查看并验证接收到的控制指令，并模拟设备端上报执行结果。

基于WebSocket方式的实体反控流程



准备事项

- 1.注册数字孪生服务DTService（以下简称DTS）账户
- 2.准备APIFox，用于模拟接收设备反向控制的指令和数据上报。
- 3.已成功创建需要进行反控的产品“温湿度传感器”。请参考[HTTP协议实体接入](#)
- 4.已成功创建需要进行反控的实体。

定义产品Profile

需在产品Profile中定义实体支持的服务。服务执行后，根据使用场景选择结果返回方式：

- 支持通过属性上报或事件上报的方式，将执行结果反馈到DTS中。采用此方式，需定义对应的属性或事件。
- 支持通过HTTP或RedisMQ的方式，将执行结果转发给第三方平台；

在产品Profile所有编辑完成后，需点击**发布**。发布后的产品Profile才能被DTS识别。智能插座的数据格式为“BV JSON”，示例默认都使用BV JSON进行数据交互。无论数据格式是BV JSON或透传/自定义流程类似，BV JSON无需写消息解析脚本。

定义产品Profile（服务）

1. 进入实体管理模块，选择产品。
2. 选择“温湿度传感器”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为**服务**。
4. 输入功能名称为“重启复位”，设置标识符为resetDeviceWS。
5. 服务地址选择“DTS 内置服务”，显示接口地址“/{productid}/{identifier}”。
6. 根据“重启复位服务”服务设置输入参数 deviceid 。
7. 根据需要设定选择结果返回方式为属性上报or事件上报。

* 功能类型

服务

* 功能名称

设备重启复位

* 标识符

resetDevice

服务地址 ⓘ

本地Postman服务 https://3cb98ad9-2243-4c27-b0c1-c5bb0f8af23f.mock.pstl

查看完整服务地址

接口

POST

/rest

调用方式

同步

异步

* 结果返回方式 ⓘ

属性上报

输入参数

参数名称: 设备编号

编辑 | 删除

+ 增加参数

输出参数

+ 增加参数

描述

请输入描述

0/100



图1：采用属性上报返回结果

* 功能类型

服务

* 功能名称

设备重启复位

* 标识符

resetDevice

服务地址 ⓘ

本地Postman服务 https://3cb98ad9-2243-4c27-b0c1-c5bb0f8af23f.mock.pstl

[查看完整服务地址](#)

接口

POST /rest

调用方式

☐ 同步 ☒ 异步

* 结果返回方式 ⓘ

事件上报

目标事件 ⓘ

获取重启状态

输入参数

参数名称: 设备编号

编辑 | 删除

+ 增加参数

输出参数

+ 增加参数

描述

请输入描述

0/100

图2：采用事件上报返回结果

同步方式为立即返回结果，异步方式可有延迟。DTS会携带回调参数发送到服务。

方式1：结果返回方式为属性

定义产品Profile（属性）

- 1. 进入实体管理模块，选择产品。
- 2. 选择“智能插座”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
- 3. 点击编辑草稿，选择添加功能，选择功能类型为**属性**。
- 4. 输入功能名称为“重启状态”，设置标识符为 status 。

* 功能类型

属性

* 功能名称

重启状态

* 标识符

status

* 数据类型

bool(布尔型)

* 布尔值

false - false

true -

true

* 读写类型

只读

读写

描述

用于接收重启复位服务的结果反馈

15/100

5. 确定保存属性。

方式2：结果返回方式为事件

定义产品Profile（事件）

1. 进入实体管理模块，选择产品。
2. 选择“智能插座”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为**事件**。
4. 输入功能名称为“获取重启状态”，设置标识符为getResult。
5. 设置输出参数 `status`。

* 功能类型

事件

* 功能名称

获取重启状态

* 标识符

getResult

* 事件类型

信息

告警

* 消息质量

默认

一级

二级

输出参数

参数名称: 重启状态

编辑 | 删除

+ 增加参数

操作

☐ 持久化存储

描述

使用事件接收重启服务的反馈结果

15/100

6. 确定保存事件。

配置消息解析脚本

仅需为使用透传/自定义协议的产品实体配置。

消息解析脚本示例

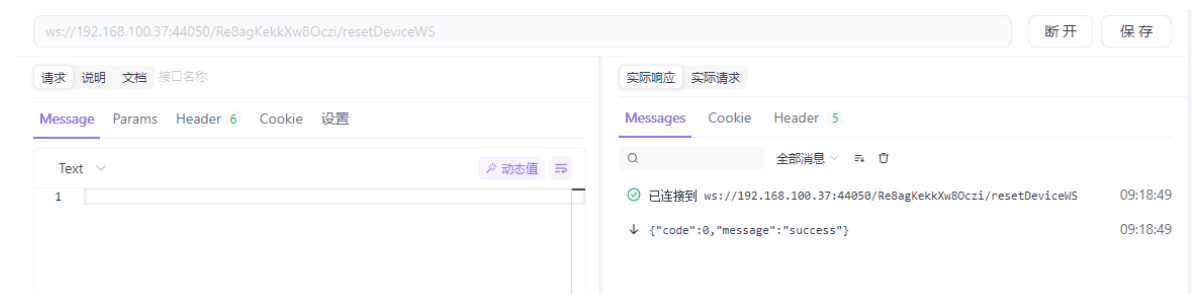
```
/**
 * 将透传/自定义数据格式转换为DTS平台的BV JSON数据格式，实体注册、属性、事件上报到DTS时调用。
 * @param {JSONObject} body - 入参
 * @param {JSONObject} body._bv_callback_message - 服务回调参数（回调时携带）
 *
 * @param {string} body._bv_callback_message.productId - 产品编号
 * @param {string} body._bv_callback_message.deviceId - 实体编号
 * @param {string} body._bv_callback_message.identifier - 事件的唯一标识符（回调方式为事件上报时携带）
 * @param {JSONObject} body._bv_callback_message._bv_service_callback - 服务调用参数
 * @param {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * @param {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识符
 * @param {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * @param {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * @param {number} flag - 入参类型标识，0=实体注册，1=属性上报，2=事件上报
 * @returns {JSONObject} 出参，BV JSON数据格式，不能为空
 */
function rawDataToProtocol(body, flag) {
  var inputData = body;
  var jsonObj = {};
  if (flag === 0) {
    /**
     * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
     * @returns {string} return.deviceName - 实体名称（最大长度255），必传
     * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
     * @returns {number} return.deviceType - 实体类型，0=动态资源，1=静态资源，必传
     * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
     * @returns {string} return.version - 版本号（默认default），非必传
     * @returns {JSONObject} return.payload - 扩展数据，用于更新实体属性值，数据基于产品 Profile 定义的属性过滤（已定义属性对应的数据保留，未定义属性对
    */
    jsonObj['deviceId'] = inputData.guid;
    jsonObj['deviceName'] = inputData.name;
    jsonObj['deviceStatus'] = inputData.status;
    jsonObj['payload'] = inputData;
    return jsonObj;
  }
  if (flag === 1) {
    /**
     * @returns {string} return.deviceId - 实体编号，需保证全局唯一性，必传
     * @returns {string} return.deviceSecret - 实体签名（不传时将系统匹配），非必传
     * @returns {string} return.deviceName - 实体名称（最大长度255），非必传
     * @returns {number} return.deviceStatus - 实体状态，0=禁用，1=启用（默认），非必传
     * @returns {string} return.location - 实体位置（默认0 0 0），格式：纬度 经度 高程，非必传
     * @returns {number} return.timestamp - 上行数据传输时间戳。格式为数值，值为从UTC 1970-01-01 00:00:00开始到当前时刻的毫秒数，非必传
    */
  }
```



建立WebSocket连接（调试步骤，若已有真实设备客户端可跳过）

使用APIFox模拟WebSocket客户端，模拟接收实体反向控制的服务“温度调节服务”的通知。模拟接收设置某个冰箱设备的温度调节指令。

1. 点击新建一个WebSocket接口，命名为“温湿度传感器”。
2. 在连接地址中输入完整的ws地址。如ws地址为“ws://192.168.100.37:44050/Re8agKekkXw8Oczl/resetDeviceWS”。**注意：**“/{productid}/{identifier}”需使用正确的产品ID `productid` 服务标识符 `identifier` 替换。
3. 点击“连接”，查看是否能正确连接到DTS内置的WebSocket服务端。
4. 连接成功，等待DTS发送需执行服务的通知。



创建任务

1. 在任务管理页面，创建一个任务名为“温度调节服务”的场景。

任务逻辑

每个工作日晚上18点 触发 执行任务 重启复位服务

1. 创建完成后，进入编辑页面。根据任务逻辑编辑对应的规则。设定触发条件为每个工作日18点，添加执行任务类型为“服务调用”，选择温度传感器产品下的某一实体或全部实体，执行服务“重启复位服务”。

温湿度定时重启

保存

定时重启温湿度传感器

如果满足一下任一触发条件

+ 添加触发条件

定时触发

每天

18:00:00

选择器

且满足以下全部生效条件

+ 添加生效条件

就执行以下任务

+ 添加执行任务

1. 设备控制-服务调用

≡

温湿度传感器 > 全部设备

执行服务

设备重启复位

参数配置

设置服务调用的参数

1. 设置服务调用的参数，`deviceid` 使用插入变量的方式，设置其值为温湿度传感器下的当前实体（执行此任务的实体）。

参数配置

×

配置服务调用时对应的输入参数的参数值,参数值可从多种来源中选择。请根据服务参数数据类型匹配正确的参数,系统将自动获取该参数值。

服务调用

设备重启复位

输入参数 选择参数值进行配置，建立映射

text(字符串)

设备编号

当前实体.产品编...

当前实体.产品编号

取消

确定

1. 保存任务，启动任务。手动执行任务，进行任务的检查。
2. 前往任务-执行日志，查看任务的执行情况。

APIFox查看请求

在APIFox的“智能冰箱”WebSocket由本套最早不接收到来自DTS的请求执行服务的数据

by callback message 发送的同步

在API文档中可以看到，WebSocket中消息体直接收到来自设备端的消息并执行服务的响应。消息体中包含回调参数。从中可获取到结果返回的方式 `_bv_callback_method` 以及结果返回地址 `_bv_callback_url`。请求体中还有执行此次服务的输入参数。



```
1  }
2      "_bv_callback_message": {
3          "_bv_service_callback": {
4              "identifier": "resetDeviceWS",
5              "requestId": "439e4ea1cf5b48c1b33ad5f5c0bc9654",
6              "version": "default",
7              "timestamp": 1752112085047
8          },
9          "productId": "Re8agKekkXw8Oczi",
10         "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372"
11     },
12     "_bv_callback_method": "http",
13     "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
14     "_bv_callback_url": "http://192.168.100.38:48083/
15     bv-universal-gateway/api/accessGateway/rawDataToProtocol/
    Re8agKekkXw8Oczi/0"
```

若服务端并未收到来自DTS的服务请求，请检查服务定义以及目标服务与DTS之间的网络是否通畅。

APIFox模拟设备端上报

1. 根据HTTP协议的HTTP协议实体接入(BV JSON数据格式)-属性上报接口、HTTP协议实体接入(BV JSON数据格式)-事件上报接口，在APIFox中编写返回结果。
2. 接口地址填写 `_bv_callback_url` 的值，请求体中 `productId` `identifier` `deviceId` 均来自 `_bv_callback_message`。
3. 接口准备完毕，点击发送。

属性上报发送请求体示例

```
{
  "method": 0,
  "productId": "MNGRcyIFFmLxxvJh",
  "deviceId": "407ddc2c-1860-4afe-a26a-cfb06228016c",
  "deviceStatus": 0,
  "payload": {
    "status": true
  }
}
```

事件上报发送请求体示例

```
{
  "method": 1,
  "productId": "MNGRcyIFFmLxxvJh",
  "deviceId": "407ddc2c-1860-4afe-a26a-cfb06228016c",
  "identifier": "getResult",
  "deviceStatus": 0,
  "payload": {
    "status": true
  }
}
```

若数据格式为透传/自定义，则需根据HTTP协议实体接入(透传/自定义)-属性上报接口、HTTP协议实体接入(透传/自定义)-事件上报接口，在APIFox中编写返回结果。

返回服务结果时，尽量携带回执参数。在透传/自定义数据格式下，可根据回执参数进行特殊的处理逻辑。

属性上报发送请求体示例

```
{
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "status": 0,
  "resetStatus": true,
  "_bv_callback_message": {
    "_bv_service_callback": {
      "identifier": "resetDevice",
      "requestId": "0d6b56b5fafd4510bf1c4339a137e91c",
      "version": "default",
      "timestamp": 1752051738967
    },
    "productId": "Re8agKekkXw8Oczl",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372"
  }
}
```

事件上报发送请求体示例

```
{
  "deviceType": "SmartRefrigerator",
  "guid": "970bdf41-7818-4718-b00e-eb14a08b6372",
  "notificationType": "getResult",
  "content": {
    "status": true
  },
  "timestamp": 1752053423202,
  "_bv_callback_message": {
    "identifier": "getResult",
    "_bv_service_callback": {
      "identifier": "resetDevice",
      "requestId": "662f7c9a02084672ad02d3e650776cca",
      "version": "default",
      "timestamp": 1752053377098
    },
    "productId": "Re8agKekkXw8Oczl",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372"
  }
}
```

结果验证

属性上报的结果可以在实体状态中直接查看，也可以经过[订阅管理](#)将向DTS订阅实体的属性变化 and 事件发生的数据，订阅端收到数据之后进行可视化处理便能对设备进行实时数据监控。DTS数据孪生架构提供快速接入查看DTS设备数据的平台功能[低代码设计器](#)（无需订阅），详细可视化功能可查看[《BigView低代码设计器教程》](#)

基于WebSocket方式进行实体反控

组合实体

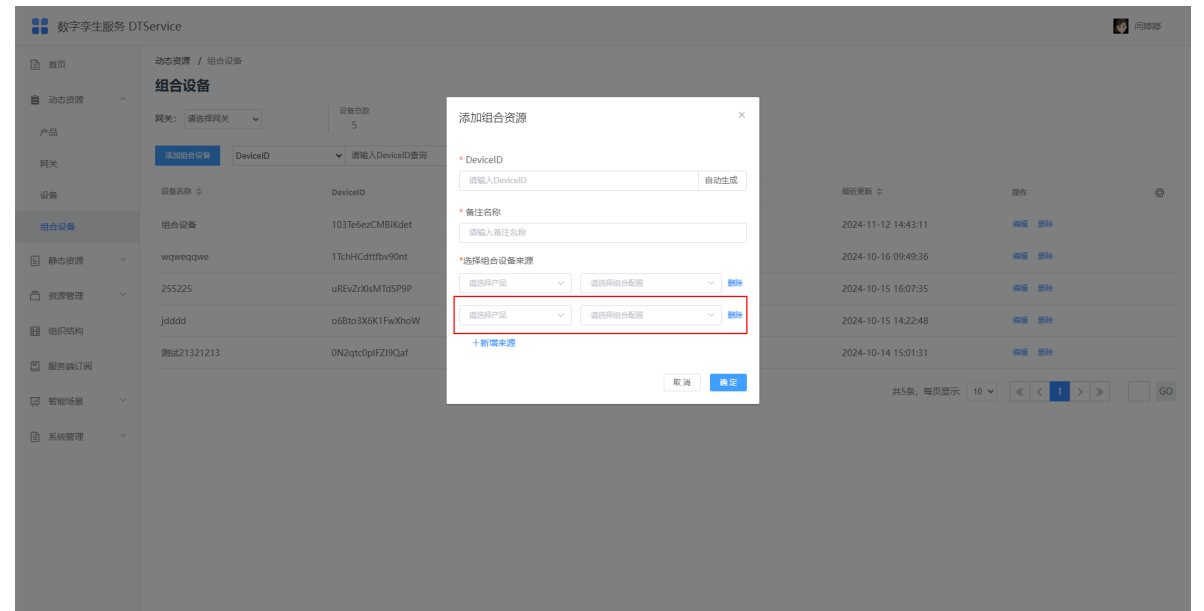
组合实体

创建组合实体

组合实体是一种特殊的虚拟实体，通常用于将多个不同实体的属性、事件、服务组合成一个统一的数据接口，且无需为每一个组合重新定义产品Profile。其上级关联到系统预置的 Dumb 网关。

[NOTE] 关于组合实体的概念机制及使用限制，请参考[概念定义 - 组合实体](#)。

选择【添加组合实体】，输入组合实体deviceID以及实体名称。系统会默认生成deviceSecret。组合实体的创建的要点在于从各个产品和已有实体下选择所需的属性、事件、服务组合成一个新的虚拟实体。



说明：1.属性，选择需要的属性字段；属性上报将默认使用组合上报的方式。 2.事件，选择需要的事件进行组合。 3.服务将按照成员实体的服务进行管理 with 调用。

编辑组合实体

在组合实体页面选择需要编辑查看的组合实体，点击"编辑"进入组合实体详情页面。

实体名称 测试组合实体2 复制

DeviceID 6QyqLYLQ7s3P8tjU 复制

DeviceSecret ***** 查看

组合实体信息

实体组合

组合实体详情页

服务订阅页

组合实体信息

编辑

实体名称	测试组合实体2	DeviceID	6QyqLYLQ7s3P8tjU	创建时间	2025-03-27 16:01:15
DeviceGuid	62b5d08f53874ae0				

成员实体信息

DeviceID	所属产品	实体名称
6F5D902833F39CF075A25E0A4945E533	8V相机	室外球机01

组合实体

属性	当前设备截图地址 可选	deviceImage	text(字符串)	ge/20250325/c15641995.jpg	2025-03-24 11:00:01
属性	服务请求编号 可选	requestId	text(字符串)	d0f887851fb94652bb63925757176a08	2025-03-13 18:00:08
属性	扩展数据 自定义	extendData	text(字符串)	("NetworkState": "1")	2025-03-27 09:50:24
属性	设备编号 自定义	deviceGuid	text(字符串)		
属性	设备类型 自定义	deviceType	enum(枚举型)	0	2025-03-27 09:50:24
属性	设备名称 自定义	deviceName	text(字符串)		

组合事件

即便在组合实体时选择了不同产品下实体的事件，初始进入组合事件时也是空的。需要将选择的实体事件重新组合成新的组合事件。

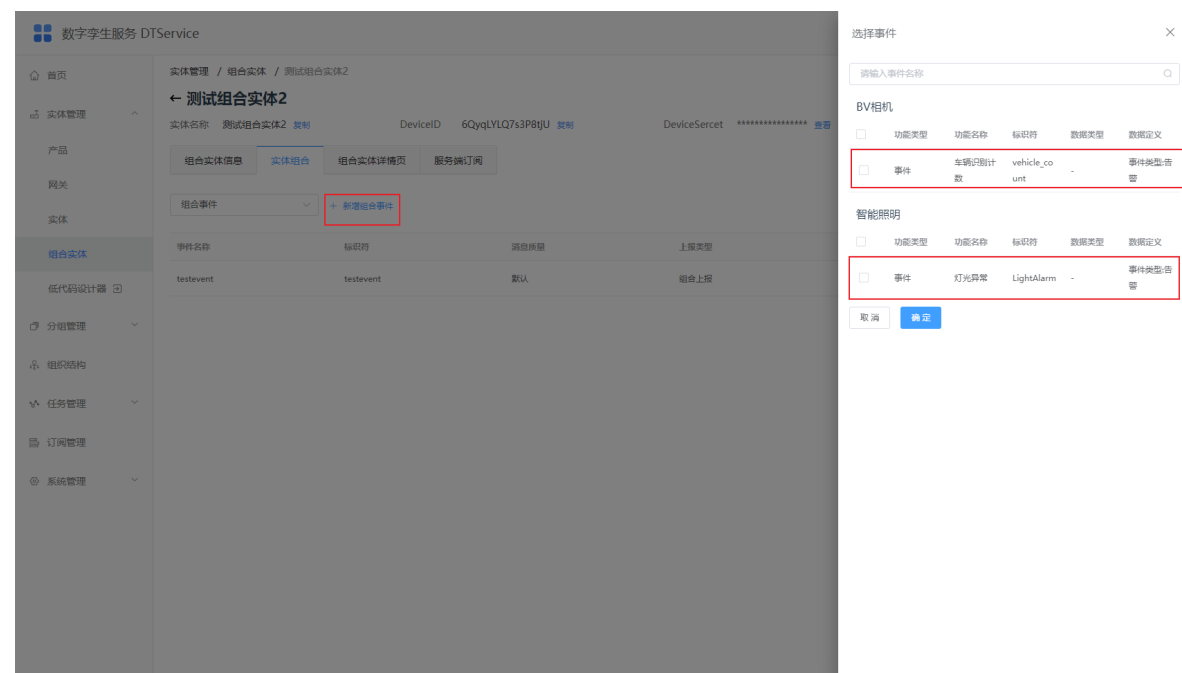
步骤：

1. 选择【添加组合事件】；

2. 在已有成员实体的事件列表中，选择需要单一个或多个事件组合成为一个新的组合事件；

3. 定义组合事件的名称、标识符、上报类型等内容。

4. 不需要的事件输出参数可以删除，但不支持新增。

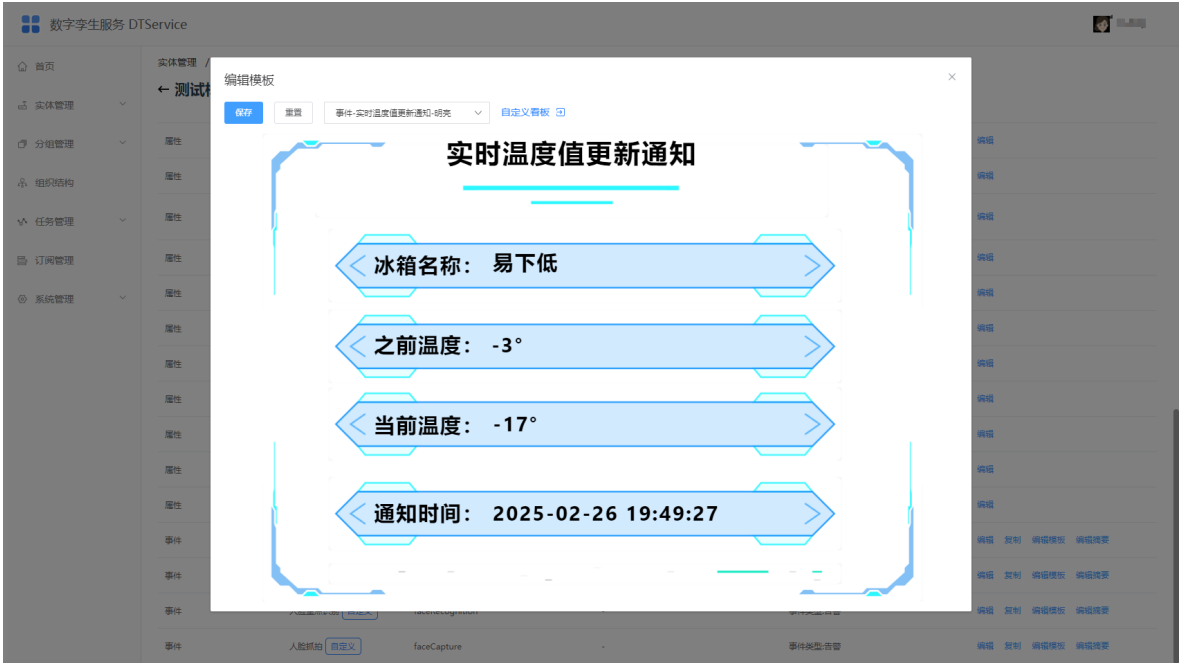


参数	描述
功能名称	组合事件的名称。符合系统命名规范。
标识符	组合事件在实体维度的唯一标识符（ <code>identifier</code> ）。
事件类型	事件级别，分为“信息”与“告警”。详细释义请参考 概念定义 - 事件 。
消息质量	消息传递的服务质量级别（QoS，支持0或2）。
上报方式	组合事件专属的上报类型： <ul style="list-style-type: none">• 成员上报：当任一成员实体触发事件时，仅上报该成员实体的输出参数；

参数	<div>• 组合上报：成员触发事件时合并上报所有参数，未触发的参数使用最新的系统缓存数据。</div>
输出参数	组合事件的出参，由关联的各成员实体的事件输出参数组合而成（不可修改成员参数）。
描述	对该组合事件的补充说明或备注，限100字。

事件模板

事件模板是展示**事件**数据的一个可视化页面，可以展示以及渲染到第三方平台，进行静态页面展示。选择看板作为事件展示时的模板样式，可通过低代码设计器绘制看板。将选择的看板保存后就成为该事件的模板。**说明**：将用来实时渲染展示，请确保输出参数**identifier**与看板使用的数据集**identifier**参数一致。不添加模板则不会进行页面渲染直接发送Json数据，默认发送模板链接以及json数据；

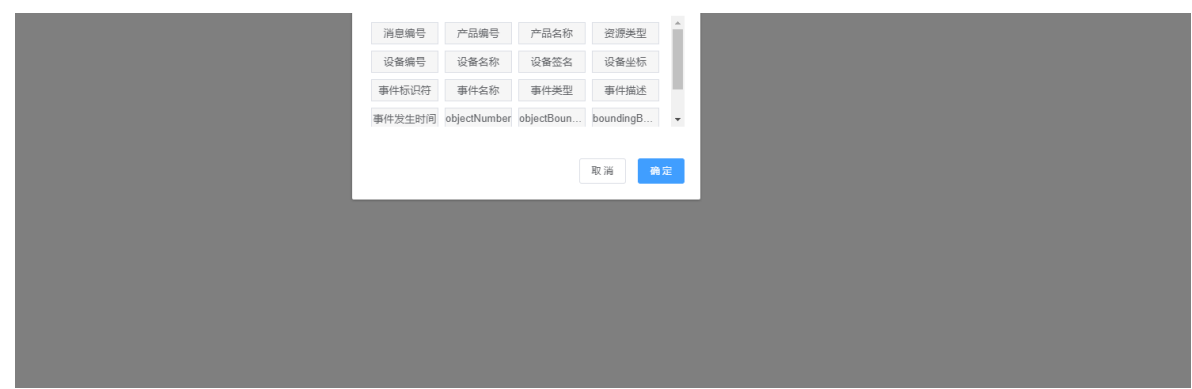


事件摘要

事件摘要是通过文本的形式简要明了的描述发生的事件。可以自行输入文本内容，也可以通过事件输出参数拼装组成。在发送事件摘要到通知中心时，会将数据值装载到摘要中。

组合上报时采用组合事件摘要，成员上报时使用成员事件摘要。





订阅关系

当前组合实体存在订阅时，可查看到所有订阅的信息。

实体管理 / 组合实体 / 测试组合实体2

← 测试组合实体2

实体名称 测试组合实体2 [复制](#) DeviceID 6QyqLYLQ7s3P8tjU [复制](#) DeviceSecret ***** [查看](#)

组合实体信息 实体组合 组合实体详情页 服务端订阅

订阅名称	订阅类型	订阅方式	消息质量	创建时间	操作
暂无数据					

分组管理

分组管理

DTS提供实体分组功能支持跨产品分类实体，您可以将实体按照一定的规则或特性（例如功能、地理位置等）进行划分为不同的资源。不同的资源又可以分组形成不同的资源组。分组管理以便进行统一管理和控制，提高管理的效率和实体的利用率。本文介绍实体分组应用场景，以及如何创建和管理分组及分组下设备。

应用场景

实体分组可以按照不同的维度进行划分，例如：

功能 将实体按照其功能特性进行分组，例如将所有摄像设备分为一个资源，将所有传感器设备分为一个资源等。再将实体按照应用领域特性进行分组，例如将摄像设备的资源、传感器的资源分为一个消防资源组。

地理位置 将实体按照其地理位置进行分组，例如将同一地区或者同一建筑物内的实体分为一个资源，方便对该地区或者建筑物进行管理和监控。再将同一省份不同地区的资源分为一个资源组。

用途 将实体按照其业务用途进行分组，例如将所有安全设备分为一个资源，将所有生产设备分为一个资源等。再按照实体分管的部门，将安全设备的资源、生产设备的资源分为一个安全生产资源组。

业务流程

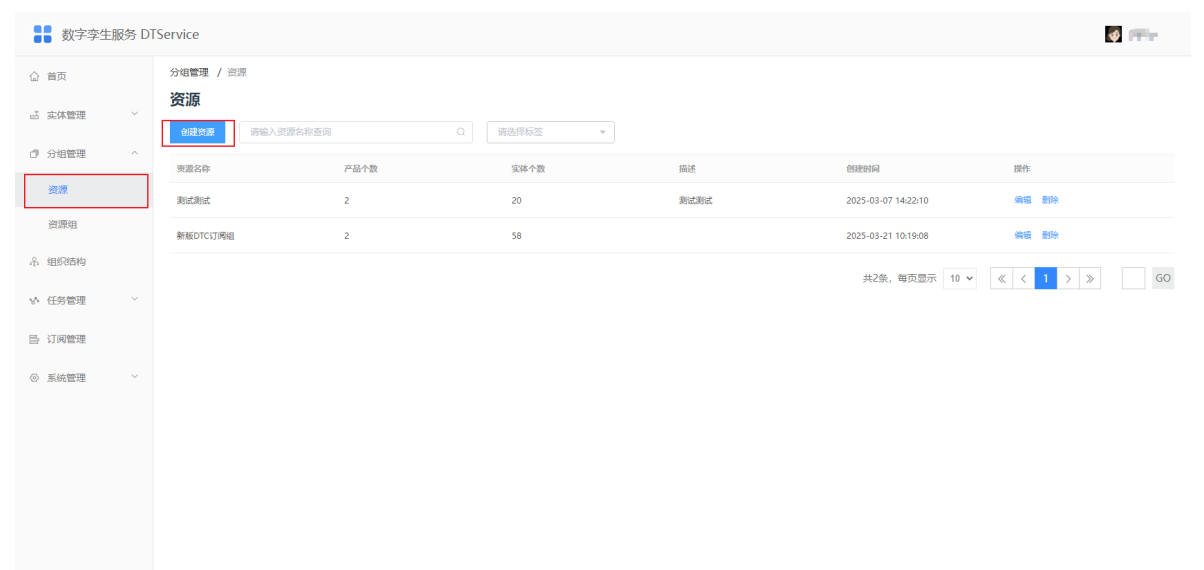
将实体按照其在业务流程中的角色进行分组，例如将所有输入实体分为一个资源，将所有输出实体分为一个资源等。还可以将输入实体资源、输出实体资源、辅助材料资源分为一个业务流程组。

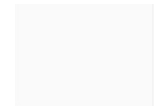
设备分组后，可在DTS服务中应用：

数据订阅：订阅者创建订阅时，可以选择按照设备分组进行订阅。

创建资源

进入分组管理选择资源页面，点击【创建资源】。输入资源名称和资源描述便可快速创建一个资源。





编辑资源

选择资源进入资源详情页，可查看资源的基本信息、资源列表、订阅关系的信息。

分组管理 / 资源 / 测试测试

← 测试测试

资源名称 测试测试 复制 实体数 20 产品数 2

资源信息

资源列表

服务端订阅

资源信息 [编辑](#)

资源名称	测试测试	产品总数	2	创建时间	2025-03-07 14:22:10
ResourceId	8c45c7f0225447bb	实体总数	20	添加人	coco
资源描述	测试测试				

标签 [编辑](#)

标签:

资源基本信息

查看和编辑的资源基本信息，资源包含的实体数量、产品数量等内容。

分组管理 / 资源 / 测试测试

← 测试测试

资源名称 测试测试 复制 实体数 20 产品数 2

资源信息

资源列表

服务端订阅

资源信息 [编辑](#)

资源名称	测试测试	产品总数	2	创建时间	2025-03-07 14:22:10
ResourceId	8c45c7f0225447bb	实体总数	20	添加人	coco
资源描述	测试测试				

标签 [编辑](#)

标签:

资源标签

资源标签用于对资源进行分类检索。资源标签支持“可选择 + 可输入”的组合模式，支持两级标签管理，既可以从下拉列表中快速选择已有的分组或标签，也允许自定义输入新标签。

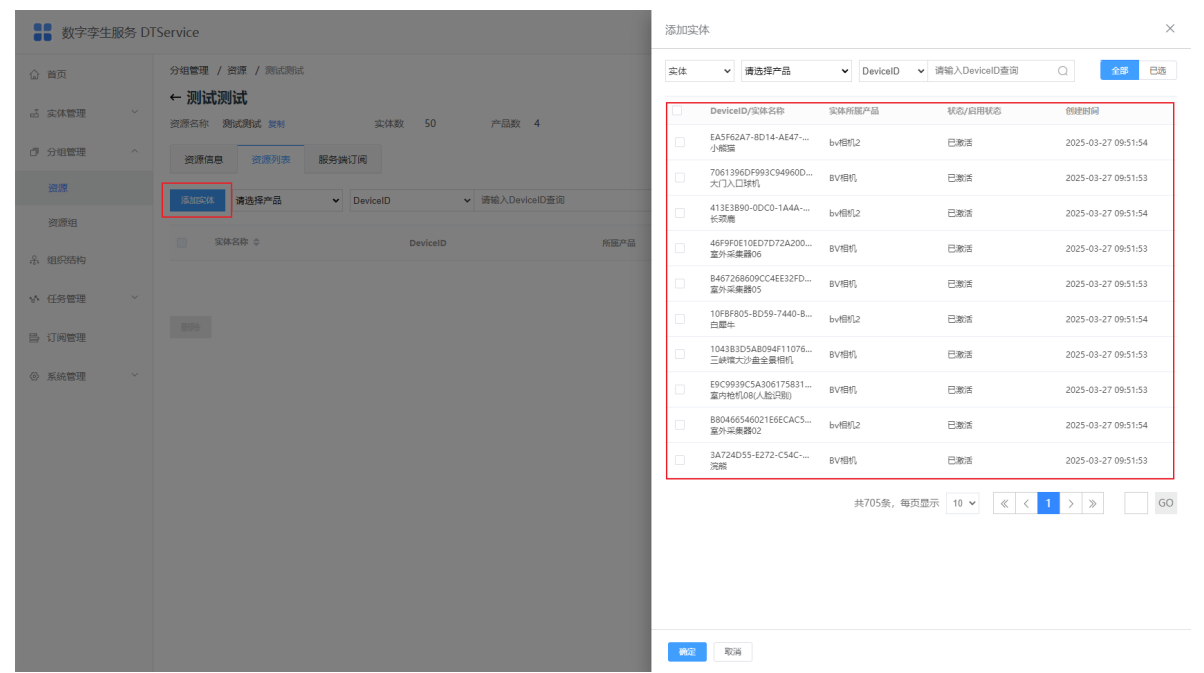
参数	说明
标签分组 (一级标签)	标签的类别或属性组，支持快速选择已有分组或自定义输入创建（如 <div>Project</div> 、 <div>Usage</div> ）。
标签 (二级标签)	具体标签值，支持快速选择已有标签或自定义输入创建（如 <div>SmartFactory</div> 、 <div>Monitoring</div> ）。

资源列表

查看资源下的实体列表，可将实体划分进行已分组的资源中进行管理。

添加实体到资源下

步骤：1.选择【添加实体】，进入实体选择页面； 2.选择需要加入到资源下的实体，选择【确定】； 3.实体成功加入到资源列表中。



删除资源中的实体

在资源列表中选中实体，点击【删除】，在删除提示中选择确认既可以删除实体。

从资源中删除实体并不会真正的从DTS中删除实体。

订阅关系

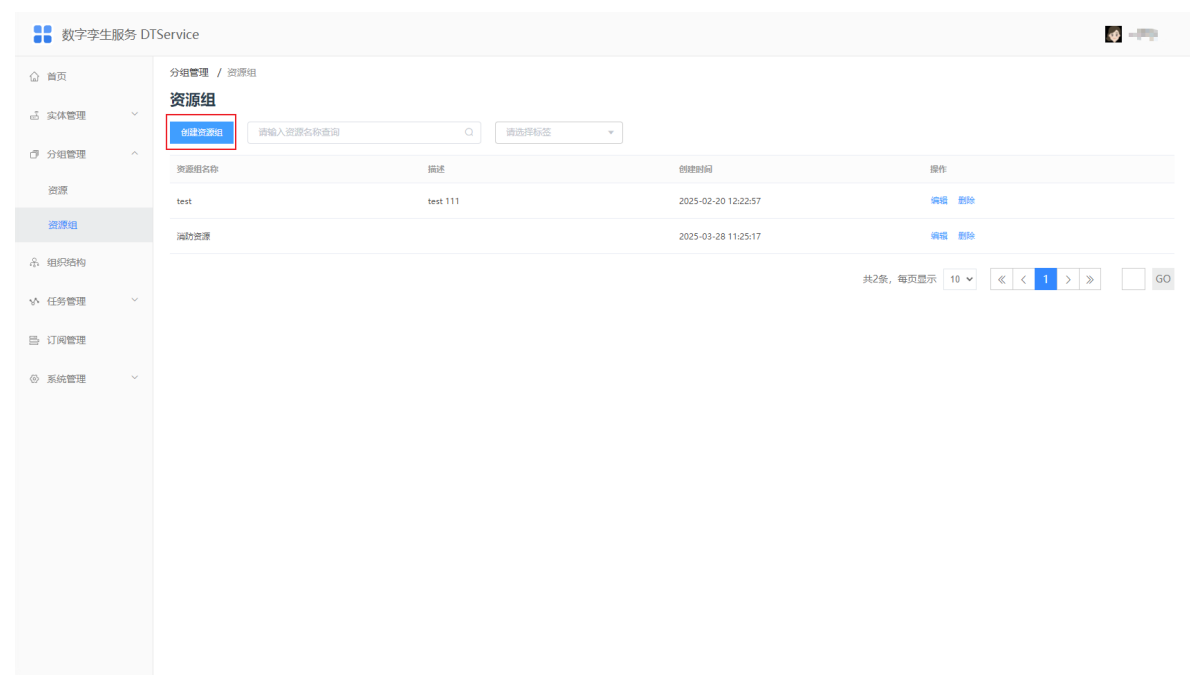
当前资源存在订阅时，可查看到所有订阅的信息。



创建资源组

资源组是将资源按照一定的规则进行再次分组。进入分组管理选择资源页面，点击【创建资源组】。输入资源组名称和资源组

描述便可快速创建一个资源组。



编辑资源组

选择资源组进入资源组详情页，可查看资源组的基本信息、资源列表、订阅关系的信息。



资源组基本信息

查看和编辑的资源组基本信息，资源组包含的实体数量、产品数量、资源数等内容。



标签:

资源组标签

资源组标签用于对资源组进行分类检索。资源组标签同样支持“可选择 + 可输入”的组合模式与二级标签管理，既可以从下拉列表中快速选择，也允许自定义输入。

参数	说明
标签分组 (一级标签)	标签的类别或属性组，支持快速选择已有分组或自定义输入创建（如 Org、Environment）。
标签 (二级标签)	具体标签值，支持快速选择已有标签或自定义输入创建（如 Division-A、Testing）。

资源列表

查看资源组下的实体列表，可将资源划分进行已分组的资源组中进行管理。

添加实体到资源下

步骤：1.选择【添加资源】，进入资源选择页面；2.选择需要加入到资源下的资源组，选择【确定】；3.资源成功加入到资源列表中。

分组管理 / 资源组 / 消防资源

← 消防资源

资源组名称 消防资源 复制 实体数 0 产品数 0 资源数 0

资源组信息

资源列表

服务端订阅

添加资源

请输入资源名称查询

资源名称	产品个数	实体个数	描述	创建时间	操作
暂无数据					

共0条，每页显示

10

« < 1 > »

GO

删除资源中的资源

在资源列表中选中资源，点击【删除】。

从资源中删除实体并不会真正的从DTS中删除实体。

订阅关系

当前资源组存在订阅时，可查看到所有订阅的信息。

分组管理 / 资源 / 测试测试

← 测试测试

资源名称 测试测试 复制 实体数 50 产品数 4

资源信息

资源列表

服务端订阅

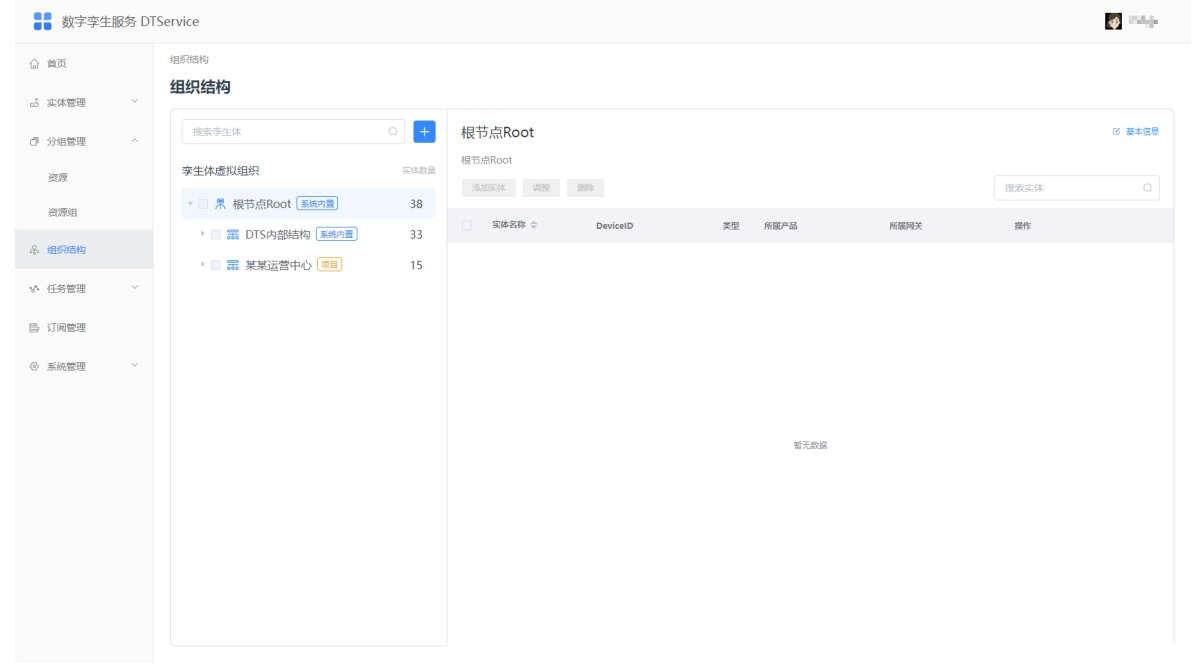
分组管理

订阅名称	订阅类型	订阅方式	消息质量	创建时间	操作
暂无数据					

组织结构

组织结构

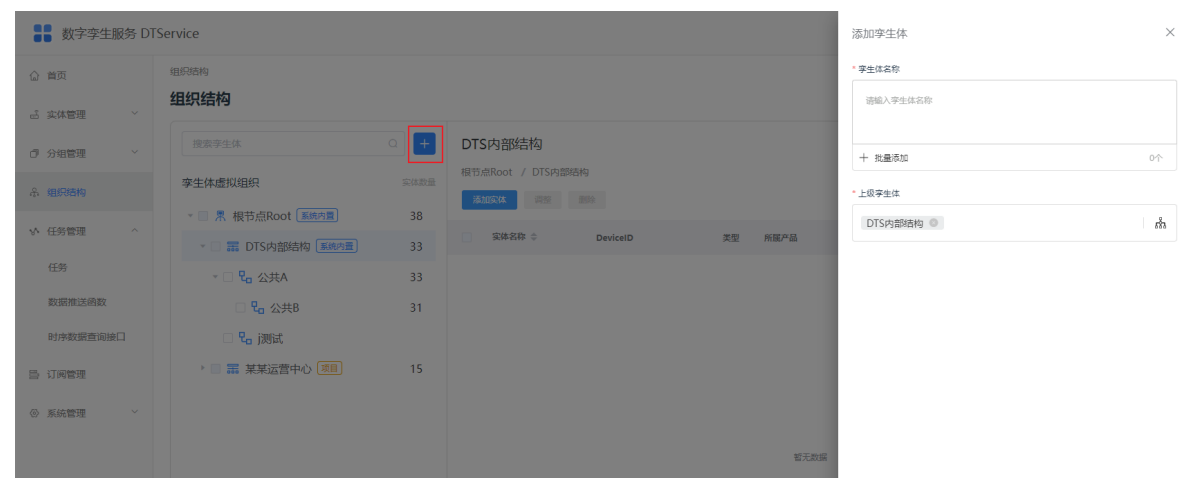
组织结构是以孪生体虚拟组织的形式组织实体，同时支持多种维度对实体进行管理。即可描述用户自己的组织结构，也可以描述用户自定义的空间组织关系。最终会形成一张孪生图，这张孪生图会整个数字孪生系统中应用。



创建孪生体

步骤：

1. 选中需要添加下级孪生体的孪生体；
2. 点击【创建孪生体】，进入创建孪生体页面
3. 输入孪生体名称；
4. 可以修改上级孪生体；
5. 输入完成，点击确定。





批量创建孪生体

通过前缀+范围+后缀的形式批量生成孪生体名称。

步骤：

1. 选中需要添加下级孪生体的孪生体；
2. 点击【创建孪生体】，进入创建孪生体页面
3. 点击【批量添加】，进入批量添加孪生体名称页面；
4. 可以修改上级孪生体；
5. 输入完成，点击确定。

批量添加

孪生体名称前缀

前缀

范围

请输入

—

请输入

序号可填写数字1~1000，汉字序号一到十，或英文字母A~Z

孪生体名称后缀

-后缀

展示结果 ?

取消

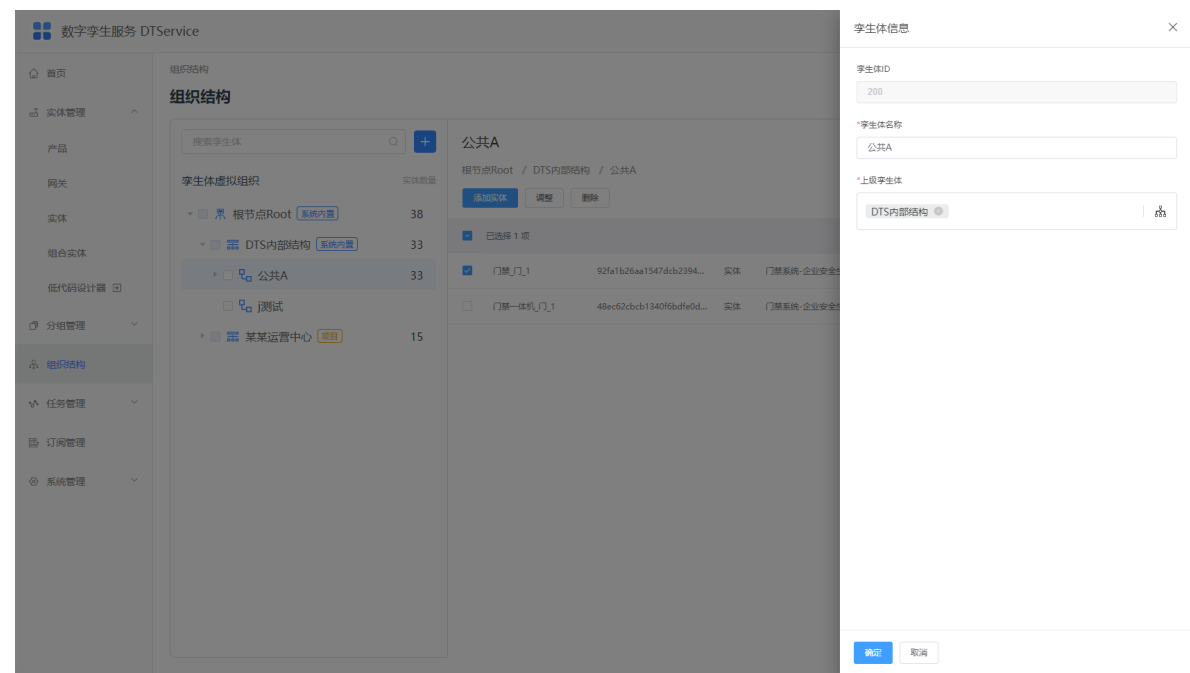
确定

编辑孪生体

编辑孪生体的基本信息，将需要此孪生体管理的实体添加到孪生体下。已在孪生体下的实体可以调整到其他孪生体下。

孪生体基本信息

进入孪生体基本信息页面，可以编辑修改孪生体名称、上级孪生体等信息。



添加实体

步骤：

1. 选中需要添加实体的孪生体；
2. 点击【添加实体】，进入实体选择页面。可以从不同的产品下添加实体。
3. 选择完成后，点击【确定】；
4. 可以查看到添加的实体在孪生体下展示；



调整实体

步骤：

1. 选中孪生体；
2. 勾选选中需要调整的实体；
3. 点击【调整】，进入选择孪生体页面；
4. 选择需要将实体移动到其下级的目标孪生体；
5. 选择完成后，点击【确定】。

5. 选择实体后，点击【删除】，
6. 提示操作成功或失败。

删除实体

选中实体删除即可。删除并不会从DTS中删除此实体，只是该实体不在此孪生体下管理。

任务管理

任务管理

通过任务，您可以：

- 创建一个自动执行的任务；
- 利用数据推送函数扩展执行任务的内容；
- 创建一个可以对外提供时序数据记录的接口；

任务

任务

任务概述

任务是云端自动化的业务规则逻辑。当特定的触发条件满足且生效条件成立时，系统会自动执行预设的动作列表，实现实体联动。

[NOTE] 关于任务定义、触发条件与执行动作的分类释义，请参考[概念定义 - 任务管理](#)。

应用场景

工业场景

可根据多个设备的温度、震动、噪声、转速等多维度数据，综合判断生产设备组的健康状态。当设备健康状态出现异常时，及时通知车间运维工程师进行及时检查修复，避免因设备故障造成的生产中断、产品不良、设备损坏等问题影响产能。

医疗健康

可通过监控一个患者的血压、心率、运动状态，结合地域、海拔、时间、温度等因素，准确判断患者是否需要医护人员介入做进一步检查，减轻患者医疗负担的同时保障个人安全。

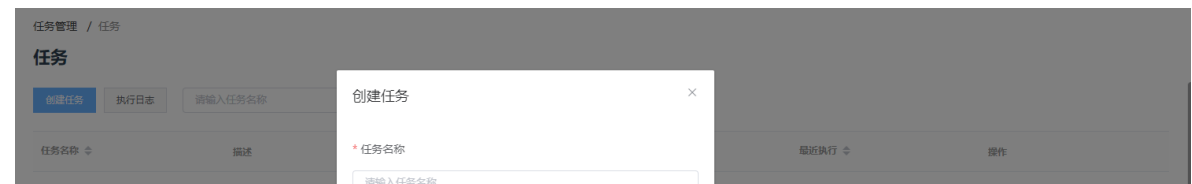
创建流程

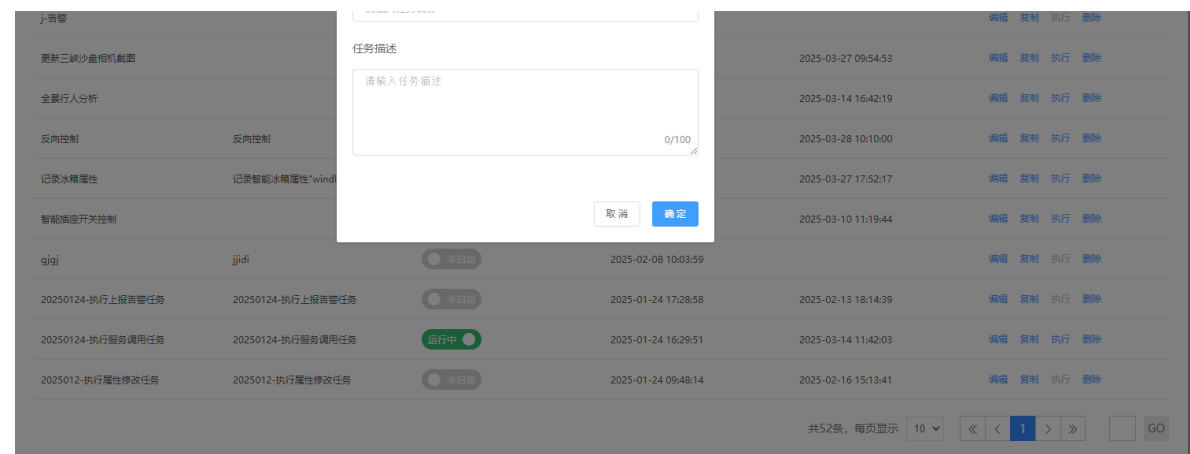


定义任务

创建任务

输入任务名称以及任务描述就可快速创建一个内容为空的任務。





任务标签

在创建或编辑任务时，可以设置任务标签用于对任务进行分类与筛选。任务标签支持“可选择 + 可输入”的组合模式与两级标签管理，既可以从下拉列表中快速选择已有的分组或标签，也允许自定义输入新标签。

参数	说明
标签分组 (一级标签)	标签的类别或分组名称，支持快速选择已有分组或自定义输入创建（如 <code>BusinessLine</code> 、 <code>Priority</code> ）。
标签 (二级标签)	具体的标签值，支持快速选择已有标签或自定义输入创建（如 <code>Factory-A</code> 、 <code>Critical</code> ）。

编辑任务

任务需配置触发条件、生效条件、执行任务，执行任务为必须配置。任务触发后，会检查是否满足生效条件，在满足条件的前提下会逐一执行任务动作。例如调用智能照明实体的开灯。

步骤：

选择【添加触发条件】，根据需要选择时间触发、属性触发、事件触发；选择【添加生效条件】，根据需要选择设备状态、时间范围；选择【添加执行任务】，选择一个或多个需要执行的任务动作；





- 1.上报告警只有在触发条件含有事件触发时才可选择。2.执行任务相关任务时，不允许任务之间存在嵌套关系。如任务A存在执行任务B的任务，任务B又存在执行任务A的任务。3.执行数据推送函数时，需要与数据推送函数的内容匹配。否则可能会存在由于缺少数据导致函数执行失败，如使用定时/手动触发执行属性数据或事件数据的时序数据记录。

运行任务

启动任务规则，使其生效。满足条件后任务即可被触发执行，任务下可查看最新执行事件触发，详细的执行结果可前往执行日志查看。

数字孪生服务 DTService

首页

动态资源

静态资源

资源管理

组织结构

服务订阅网

智能场景

场景管理

数据推送函数

时序数据查询接口

系统管理

智能场景 / 场景管理 / 执行日志

← 执行日志

全部状态

全部场景

自定义

2023-12-04 14:33:30 - 2024-12-03 14:33:30

场景名称	执行类型	执行任务	执行时间	执行状态	操作
门禁变化触发提醒	自动执行	数据推送函数 - 执行函数	2024-11-28 11:10:20	成功	详情
门禁变化触发提醒	自动执行	数据推送函数 - 执行函数	2024-11-28 11:10:20	成功	详情
门禁变化触发提醒	自动执行	数据推送函数 - 执行函数	2024-11-28 11:09:12	成功	详情
门禁变化触发提醒	自动执行	数据推送函数 - 执行函数	2024-11-28 11:09:12	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 15:07:21	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 15:07:20	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 14:54:59	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 14:40:12	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 14:38:30	成功	详情
人脸触发行人计数	自动执行	设备控制 - 服务调用、设备控制 - 服务调用	2024-11-26 14:34:40	成功	详情

共9079条，每页显示 10 < 1 > GO

执行日志

执行日志用于记录任务每次触发运行的详细过程和结果。用户可以在任务列表页，点击对应任务的“执行日志”按钮，查看每次执行的开始时间、结束时间、执行状态以及每个任务动作（如服务调用或属性修改）的成功/失败详情及返回报文。如果执行失败，可结合系统管理中的“日志服务”进行详细的排查调试。

数据推送函数

数据推送函数

数据推送函数是在安全沙箱中执行的自定义逻辑脚本。它提供了一种灵活的数据交互转发模式，支持将上行数据分发至 RedisMQ、第三方 HTTP 回调或写入时序存储。

[!NOTE] 关于数据推送函数和时序记录的核心概念，请参考[概念定义 - 数据推送函数](#)。示例应用包括：

- 1. 将产生的变化推送到内容推送到第三方平台（钉钉）中进行通知展示。如设备行为满足任务中事件和条件，将设备信息和事件内容推送到第三方平台（钉钉）中进行通知展示。
- 2. 在设备属性发生变化或事件发生时，将所需要的设备属性或事件输出参数记录或经过计算后记录下来

创建函数

定义函数的名称和函数类型。根据不同的请求类型，可以选择Redis或Http请求。时序数据记录（属性）和时序数据记录（事件）为系统提供将属性和事件信息进行永久存储的方式。数据推送函数处于**启用状态**时，才能作为任务的执行任务被选择。

数字孪生服务 DTService

首页

动态资源

静态资源

资源管理

组织结构

服务端订阅

智能场景

场景管理

数据推送函数

时序数据查询接口

系统管理

智能场景 / 数据推送函数

数据推送函数

创建函数

请输入函数名称

函数名称	描述	请求类型	启动状态	创建时间	最近更新	操作
ytt的测试函数	门禁属性在线状态摘要	时序数据记录（属性）	使用中	2024-11-22 10:27:56	2024-11-28 10:46:31	编辑 删除
车辆计数统计	1、看门狗产品定时服务调...	时序数据记录（属性）	未启用	2024-10-20 17:18:23	2024-10-29 17:21:27	编辑 删除
时序数据记录（事件）示例	TestEvent123	时序数据记录（事件）	未启用	2024-10-20 11:48:58	2024-10-29 16:38:40	编辑 删除
时序数据记录（属性）示例		时序数据记录（属性）	未启用	2024-10-20 11:36:12	2024-11-28 10:57:15	编辑 删除
Redis测试示例		Redis	未启用	2024-10-20 11:35:16	2024-10-20 11:35:16	编辑 删除
HTTP测试示例		HTTP	未启用	2024-10-20 11:34:47	2024-10-20 11:34:47	编辑 删除
数据接入	第三方网关数据接入：可修...	HTTP	未启用	2024-10-10 15:07:38	2024-11-28 10:31:16	编辑 删除
BH-抓拍统计		时序数据记录（事件）	未启用	2024-03-24 17:03:39	2024-10-08 16:56:08	编辑 删除
BH-告警-人员统计	告警统计-人员统计-事件	时序数据记录（事件）	未启用	2024-03-24 11:49:33	2024-11-10 09:39:20	编辑 删除
BH-进入统计-记录		时序数据记录（事件）	使用中	2024-03-21 05:10:10	2024-11-26 14:28:41	编辑 删除

共32条，每页显示 10 < 1 > >> GO

数据推送函数

数据推送函数提供灵活便捷的数据交互转发模式，可在场景中作为任务被执行，支持RedisMQ、Http请求以及时序数据记录。

创建

应用

1 2 3

1 2 3 4

推送函数与时序记录标签

在创建或编辑数据推送函数（包含属性/事件时序数据记录）时，可以设置标签用于对推送函数进行分类与筛选。该功能支持“可选择 + 可输入”的组合模式与两级标签管理，既可以从下拉列表中快速选择已有的分组或标签，也允许自定义输入新标签。

参数	说明
标签分组（一级标签）	标签的类别或分组名称，支持快速选择已有分组或自定义输入创建（如 <code>FunctionType</code> 、 <code>Storage</code> ）。

本文档由北京大视界科技有限公司生成- 124 -

标签 参数 (一级标签)	说明
	具体的标签值，支持快速选择已有标签或自定义输入创建（如 Notification 、 TimescaleDB ）。

编写函数

使用JavaScript编写脚本函数，定义函数需要做的内容。时序数据记录（属性）和时序数据记录（事件）需指定要存储的内容。
此功能具有一定难度，需要使用者具有一定的开发能力。

时序数据查询接口

时序数据查询接口

时序数据查询接口，是将通过时序数据记录下的数据对外提供访问的途径。通过接口请求到对应的数据结果集。可用于图表等内容的数据渲染。

创建接口

配置接口可以查询到的数据内容。以组织结构划分实体所属，设定实体可查询的时序数据记录，查询范围以及输出规则。

数字孪生服务 DTService

首页

动态资源

静态资源

资源管理

组织结构

服务端订阅

智能场景

场景管理

数据推送函数

时序数据查询接口

系统管理

智能场景 / 时序数据查询接口

时序数据查询接口

创建查询接口

请输入查询接口名称

查询接口名称	描述	启用状态	创建时间	操作
yt的时序查询接口		已启用	2024-11-26 15:46:07	编辑 查看结果集范例 删除
HQ-测试	HQ-测试	已启用	2024-11-22 17:49:45	编辑 查看结果集范例 删除
Jddddd	111222333444555666	未启用	2024-11-21 09:47:39	编辑 删除
BH-抓拍统计		未启用	2024-03-24 17:55:05	编辑 删除
BH-获取指定空间下的设备	用于实时播放	未启用	2024-03-24 16:30:38	编辑 删除
BH-告警统计	1、按payload中的eventType分组计数	未启用	2024-03-24 15:16:46	编辑 删除
BH-设备统计	统计当前孪生体以及孪生体下的门禁和...	已启用	2024-03-24 14:56:56	编辑 查看结果集范例 删除
BH-进入记录	1、查询指定空间下的统计数据 2、默认...	未启用	2024-03-24 10:30:16	编辑 删除
BH-进入统计	1、按时间分组 2、按设备分组 3、展示...	未启用	2024-03-22 14:48:26	编辑 删除
hsjy-告警模板展示		未启用	2023-05-16 11:13:02	编辑 删除

共12条，每页显示: 10 < 1 > GO

时序数据查询接口新手引导

时序数据查询接口可视化治理，运用于数据数据来源、AI算法等应用模块，通过孪生体、查询方式等参数请求时序数据，并返回结果。

创建

1 2 3 4

应用：图表

1 2 3 4

功能名称	含义
孪生体	查询指定孪生体下实体产生的时序数据
时序数据名称	查询指定的时序数据记录
时间范围	查询指定的时间范围内的时序数据记录
查询内容	查询的时序数据记录内容。 统计查询：统计时序数据记录的数量； 条件查询：查询满足条件的时序数据记录
查询方法	使用不同的查询方法对请求数据进行处理，提取出所需数据。 分组查询：对查询结果进行分组处理，支持按时序、产品、实体进行分组； 按时间排序：对查询结果按照时间的正序或逆序排列； 分页查询：查询指定页码的指定条目数的结果。

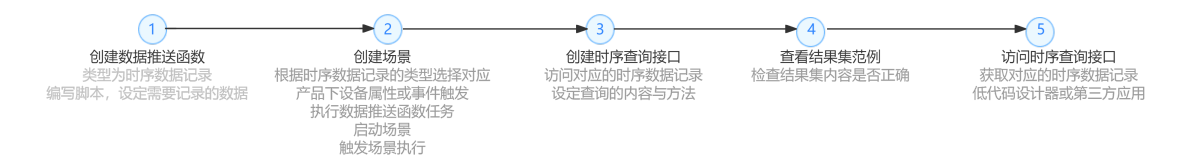
本文档由北京大视景科技有限公司生成- 126 -

启用接口

启用接口，保持接口的可访问状态。访问接口，请参考《对接开发指南》—[时序数据查询接口](#)

应用流程说明

建立和使用一个时序查询接口的完整引用流程。



结果集范例

结果集范例中包含请求接口地址、请求参数以及响应结果示例。可根据请求参数格式在请求时调整接口，修改可查询到的数据内容。可根据响应结果进行前端界面的开发。若查询到的时序记录为空可先检查执行数据推送函数的任务是否有成功执行。

实体历史属性数据

历史实体属性数据

本文将为你介绍使用DTS进行时序数据记录（属性），利用时序数据制作看板的流程。时序数据记录（事件）可参考此流程。包含以下几个流程： 创建数据推送函数、创建任务、创建时序数据接口、数据应用。

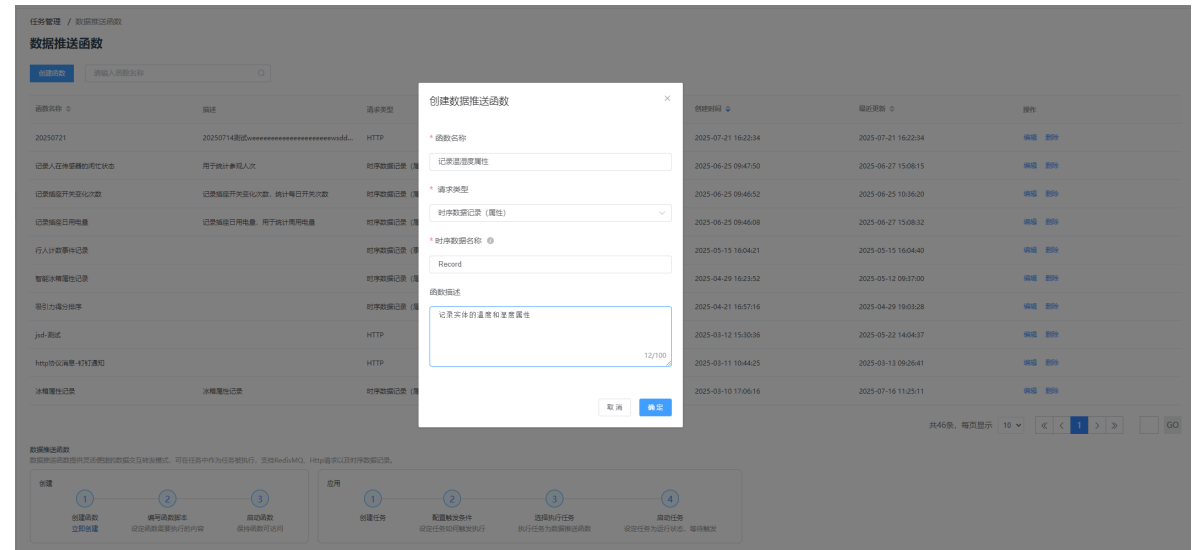
准备事项

- 1.注册数字孪生服务DTService（以下简称DTS）账户 2.准备APIFox，用于模拟设备属性、事件上报。 3.已成功创建产品”温湿度传感器“。请参考[HTTP协议实体接入\(透传/自定义\)](#)

示例

创建数据推送函数（时序数据记录）

在任务管理-数据推送函数下，创建一个名为”温湿度属性记录“的函数，请求类型需设置为**时序数据记录(属性)**。时序数据名称可自定义，用于标识时序数据,请按照函数真实含义命名。创建完成后，进入编辑脚本页面。



编辑脚本

在脚本内编辑，需要被记录下来的温湿度传感器的属性。选择记录温湿度传感器的 `temperature` `humidity` `batteryLevel` 。记录多个属性时，使用 `,` 逗号进行分割即可。如果需要对记录内容进行逻辑处理计算，可在 `sequenceAbstractCalculate`函数中编写具体的逻辑。

```
/**
 * 时序数据记录，定义需要记录的摘要参数，从当前上行结构化数据中取值。
 * @param {Array<string>} abstractParam - 摘要参数集合，当为属性上报时摘要参数标识为产品属性标识符，当为事件上报时摘要参数标识为事件输出参数标识符（
 */
var abstractParam = ["temperature","humidity","batteryLevel"];
```

```
/**
 * 将摘要数据进行二次计算
 * @param {object} body - 入参 eg: {"sourceParam": {}, "abstractParam": {}}
 * @param {object} body.sourceParam - 上行数据的固定属性，可用于摘要赋值计算等。
 * @param {string} body.sourceParam.messageId - 消息编号
 * @param {string} body.sourceParam.productId - 产品唯一标识
 * @param {string} body.sourceParam.productName - 产品名称
 * @param {string} body.sourceParam.sourceType - 资源类型 0 动态资源 1 静态资源
 * @param {string} body.sourceParam.deviceId - 设备编号
 * @param {string} body.sourceParam.deviceSecret - 设备密钥，与deviceId同时存在
 * @param {string} body.sourceParam.deviceName - 设备名称
 * @param {string} body.sourceParam.deviceStatus - 设备状态 0 未启用 1 已启用
 * @param {string} body.sourceParam.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation）
 * @param {string} body.sourceParam.identifier - 事件标识符，事件上报时存在，否则Val为空
 * @param {string} body.sourceParam.name - 功能名称，事件上报时存在，否则Val为空
 * @param {string} body.sourceParam.type - 事件类型 0 信息 1 告警，事件上报时存在，否则Val为空
 * @param {string} body.sourceParam.description - 事件描述，事件上报时存在，否则Val为空
 * @param {string} body.sourceParam.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数
 * @param {string} body.sourceParam.templateId - 消息（事件、属性详情）模板唯一标识符
 * @param {string} body.sourceParam.templateURL - 消息（事件、属性详情）模板访问链接
 * @param {string} body.sourceParam.version - 版本号。若不传入此参数，则默认default
 * @param {object} body.sourceParam.payload - 验证解析后的上行数据
 * @param {object} body.abstractParam - 摘要对象，已根据上行数据赋值
 * @returns {object} abstractParam - 摘要对象
 */
function sequenceAbstractCalculate(body) {
    return body.abstractParam;
}
```

编写完成后，保存脚本。可以使用模拟输入检查脚本内容。启用“温湿度属性记录”的函数。

时序数据记录（事件）创建与编辑一致，需注意的是时序数据记录（事件）记录的内容为事件的输出参数。

创建任务

创建任务的目的是，在属性或事件发生时，通过任务触发执行脚本函数（时序数据记录），将需要记录下来的属性或事件输出参数记录到数据库中。

1. 在任务管理页面，创建一个任务名为“记录温湿度属性”的任务。

任务逻辑

温湿度传感器全部设备发生属性上报事件 触发 执行 数据推送函数温湿度属性记录

1. 创建完成后，进入编辑页面。根据任务逻辑编辑对应的规则。设定触发条件为属性触发：温湿度传感器设备发生属性上报，添加执行任务类型为“数据推送函数”，选择数据推送函数“温湿度属性记录”。
2. 编辑完成后保存并开启任务。

记录温湿度属性

保存

请输入任务描述

如果满足一下任一触发条件

+ 添加触发条件

属性触发

温湿度传感器 > 全部设备

发生

全部属性

属性上报时

且满足以下全部生效条件

+ 添加生效条件

就执行以下任务

+ 添加执行任务

1. 数据推送函数-执行函数

记录温湿度属性

函数

模拟属性上报触发

使用APIFox模拟温湿度传感器实体属性上报。

数据应用

方式1：时序数据查询接口

创建时序查询接口

时序查询接口页面是将HTTP请求的请求体中的请求参数进行了可视化配置。页面中配置的内容实际都为查询时的请求参数。在任务管理下进入时序数据查询接口页面。选择创建一个名为“温湿度属性查询”的接口。用于查询已通过时序数据记录的温湿度传感器属性。

进行创建之前需要在组织结构下，先将温度传感器实体通过孪生体进行管理。示例将其所有温度传感器实体使用孪生体“温度传感器”进行管理。

选择查询的孪生体为“温湿度传感器”，查询的目标时序数据记录为“温湿度属性记录”，查询内容为统计查询。并将查询结果按时间倒序排列。设置完成后，点击保存并启用。

查询温湿度属性

保存

请输入接口描述

* 孪生体

☐ 递归查询所有下级孪生体

温湿度传感器

* 时序数据名称

Record(记...

共1个

☐ 按时间范围查询 示例: 2025-07-22 ~ 2025-07-23

* 查询内容

统计查询

统计时序数据记录的数量

条件查询

满足条件的时序数据记录

+添加条件

查询方法

+新增查询方法

按时间查询排序

按时间倒序

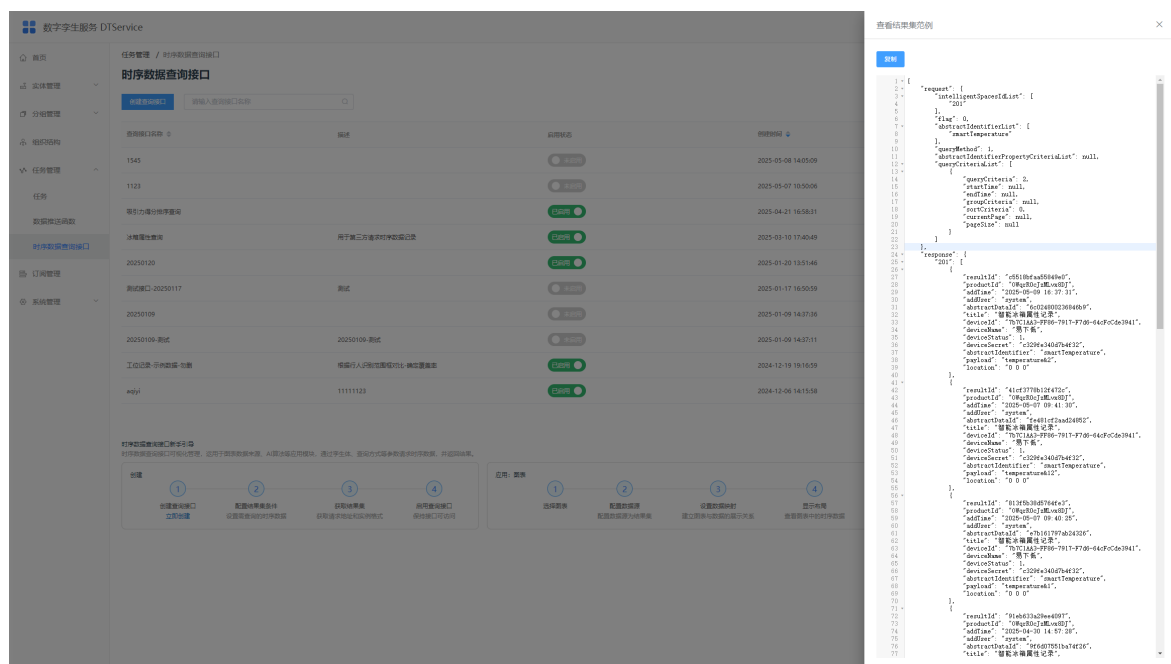
功能名称	含义
------	----

本文档由北京大视景科技有限公司生成- 131 -

功能名称 孪生体	含义 查询指定孪生体下实体产生的时序数据
时序数据名称	查询指定的时序数据记录
时间范围	查询指定的时间范围内的时序数据记录
查询内容	查询的时序数据记录内容。 统计查询：统计时序数据记录的数量； 条件查询：查询满足条件的时序数据记录
查询方法	使用不同的查询方法对请求数据进行处理，提取出所需数据。 分组查询：对查询结果进行分组处理，支持按时序、产品、 实体进行分组； 按时间排序：对查询结果按照时间的正序或逆序排列； 分页查询：查询指定页码的指定条目数的结果。

查看结果集范例

“查询温湿度属性”接口启用后，可点击查看结果集范例。结果集范例包含请求体示例和返回结果响应示例。通过查看结果集范例还能验证时序数据是否记录成功，响应结果中是否有内容。完整的接口内容请参考[时序数据查询接口](#)。



结果集范例

```
{
  "request": {
    "intelligentSpacesIdList": [
      "bf029ce259a74835"
    ],
    "flag": 0,
    "abstractIdentifierList": [
      "Record"
    ],
    "queryMethod": 1,
    "abstractIdentifierPropertyCriteriaList": null,
    "queryCriteriaList": [
      {
        "queryCriteria": 2,
        "startTime": null,
        "endTime": null,
        "groupCriteria": null,
        "sortCriteria": 0,
        "currentPage": null,
        "pageSize": null
      }
    ]
  },
  "response": {
    "bf029ce259a74835": [
      {
        "resultId": "1eb9a1b696254f32",
        "productId": "Re8agKekkXw8Oczl",
        "addTime": "2025-07-22 15:21:24",
        "addUser": "system",
        "abstractDataId": "160abc8160194695",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&28.86,humidity&57.17,batteryLevel&43",
        "location": "0 0 0"
      },
      {
        "resultId": "3239b0d4a7244229",
        "productId": "Re8agKekkXw8Oczl",
        "addTime": "2025-07-22 15:21:23",
        "addUser": "system",
        "abstractDataId": "8703096e95ef4431",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&38.78,humidity&2.31,batteryLevel&88",
        "location": "0 0 0"
      },
      {
        "resultId": "b4c3a169713d4925",
        "productId": "Re8agKekkXw8Oczl",
        "addTime": "2025-07-22 15:21:22",
        "addUser": "system",
        "abstractDataId": "8c9cb82869924438",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&14.32,humidity&24.85,batteryLevel&80",
        "location": "0 0 0"
      },
      {
        "resultId": "36889a3100044b60",
        "productId": "Re8agKekkXw8Oczl",
        "addTime": "2025-07-22 15:21:21",
        "addUser": "system",
        "abstractDataId": "2e69d7eb09b8422c",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5"
      }
    ]
  }
}
```

```

    "abstractIdentifier": "Record",
    "payload": "temperature&8.14,humidity&16.84,batteryLevel&18",
    "location": "0 0 0"
  },
  {
    "resultId": "8b042827cf5845b7",
    "productId": "Re8agKekkXw8Oczl",
    "addTime": "2025-07-22 15:21:19",
    "addUser": "system",
    "abstractDataId": "01fc627180f1424d",
    "title": "记录温湿度属性",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "deviceName": "温湿度传感器1751939165",
    "deviceStatus": 0,
    "deviceSecret": "9d4e0d3949134bd5",
    "abstractIdentifier": "Record",
    "payload": "temperature&3.2,humidity&21.94,batteryLevel&52",
    "location": "0 0 0"
  },
  {
    "resultId": "24a17b1d0c7b4529",
    "productId": "Re8agKekkXw8Oczl",
    "addTime": "2025-07-22 15:21:14",
    "addUser": "system",
    "abstractDataId": "3504cb8b8bcb43e7",
    "title": "记录温湿度属性",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "deviceName": "温湿度传感器1751939165",
    "deviceStatus": 0,
    "deviceSecret": "9d4e0d3949134bd5",
    "abstractIdentifier": "Record",
    "payload": "temperature&36.95,humidity&43.72,batteryLevel&98",
    "location": "0 0 0"
  },
  {
    "resultId": "76be272476ac4a0e",
    "productId": "Re8agKekkXw8Oczl",
    "addTime": "2025-07-22 15:18:38",
    "addUser": "system",
    "abstractDataId": "45b6f1ad43bb44db",
    "title": "记录温湿度属性",
    "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
    "deviceName": "温湿度传感器1751939165",
    "deviceStatus": 0,
    "deviceSecret": "9d4e0d3949134bd5",
    "abstractIdentifier": "Record",
    "payload": "temperature&-8.41,humidity&14.62,batteryLevel&90",
    "location": "0 0 0"
  }
]
}
```

使用APIFox创建请求查询时序数据

第三方可通过时序数据查询接口获取时序数据记录，建立自己的数据展示应用。示例使用APIFox模拟第三方发送时序数据查询请求并获得响应的过程。

- 1. 根据[时序数据查询接口](#)，在APIFox中编写时序数据查询接口的调用请求。
- 2. 接口地址填写 `http://192.168.100.38:48083/bv-microservice-digitaltwin-center/sceneLinkage/bvSpaceAbstractTemplate/findSpaceAbstractResultSet`（请求地址的IP值根据实际部署环境进行替换）的值，从结果集范例中复制 `request` 的内容至请求体中。修改查询时间范围为"2025-07-22"到"2025-07-23"。
- 3. 接口准备完毕，点击发送。

发送请求体示例

```
{
  "intelligentSpacesIdList": [
    "bf029ce259a74835"
  ],
  "flag": 0,
  "abstractIdentifierList": [
    "Record"
  ],
  "queryMethod": 1,
```

```
{
  "abstractIdentifierPropertyCriteriaList": null,
  "queryCriteriaList": [
    {
      "queryCriteria": 2,
      "startTime": "2025-07-22",
      "endTime": "2025-07-23",
      "groupCriteria": null,
      "sortCriteria": 0,
      "currentPage": null,
      "pageSize": null
    }
  ]
}
```

1. 获得DTS返回的响应结果。参数的详细含义可参考[时序数据查询接口](#)。

响应结果示例

```
{
  "code": 0,
  "message": "success",
  "info": {
    "request": {
      "intelligentSpacesIdList": [
        "bf029ce259a74835"
      ],
      "flag": 0,
      "abstractIdentifierList": [
        "Record"
      ],
      "queryMethod": 1,
      "abstractIdentifierPropertyCriteriaList": null,
      "queryCriteriaList": [
        {
          "queryCriteria": 2,
          "startTime": "2025-07-22",
          "endTime": "2025-07-23",
          "groupCriteria": null,
          "sortCriteria": 0,
          "currentPage": null,
          "pageSize": null
        }
      ]
    },
    "response": {
      "bf029ce259a74835": [
        {
          "resultId": "1eb9a1b696254f32",
          "productId": "Re8agKekkXw8Oczi",
          "addTime": "2025-07-22 15:21:24",
          "addUser": "system",
          "abstractDataId": "160abc8160194695",
          "title": "记录温湿度属性",
          "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
          "deviceName": "温湿度传感器1751939165",
          "deviceStatus": 0,
          "deviceSecret": "9d4e0d3949134bd5",
          "abstractIdentifier": "Record",
          "payload": "temperature&28.86,humidity&57.17,batteryLevel&43",
          "location": "0 0 0"
        },
        {
          "resultId": "3239b0d4a7244229",
          "productId": "Re8agKekkXw8Oczi",
          "addTime": "2025-07-22 15:21:23",
          "addUser": "system",
          "abstractDataId": "8703096e95ef4431",
          "title": "记录温湿度属性",
          "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
          "deviceName": "温湿度传感器1751939165",
          "deviceStatus": 0,
          "deviceSecret": "9d4e0d3949134bd5",
          "abstractIdentifier": "Record",
          "payload": "temperature&38.78,humidity&2.31,batteryLevel&88",
          "location": "0 0 0"
        },
        {
          "resultId": "b4c3a169713d4925",
          "productId": "Re8agKekkXw8Oczi",
          "addTime": "2025-07-22 15:21:22",
          "addUser": "system"
        }
      ]
    }
  }
}
```

```

        "abstractDataId": "8c9cb82869924438",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&14.32,humidity&24.85,batteryLevel&80",
        "location": "0 0 0"
    },
    {
        "resultId": "36889a3100044b60",
        "productId": "Re8agKekkXw8Oczi",
        "addTime": "2025-07-22 15:21:21",
        "addUser": "system",
        "abstractDataId": "2e69d7eb09b8422c",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&8.14,humidity&16.84,batteryLevel&18",
        "location": "0 0 0"
    },
    {
        "resultId": "8b042827cf5845b7",
        "productId": "Re8agKekkXw8Oczi",
        "addTime": "2025-07-22 15:21:19",
        "addUser": "system",
        "abstractDataId": "01fc627180f1424d",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&3.2,humidity&21.94,batteryLevel&52",
        "location": "0 0 0"
    },
    {
        "resultId": "24a17b1d0c7b4529",
        "productId": "Re8agKekkXw8Oczi",
        "addTime": "2025-07-22 15:21:14",
        "addUser": "system",
        "abstractDataId": "3504cb8b8bcb43e7",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&36.95,humidity&43.72,batteryLevel&98",
        "location": "0 0 0"
    },
    {
        "resultId": "76be272476ac4a0e",
        "productId": "Re8agKekkXw8Oczi",
        "addTime": "2025-07-22 15:18:38",
        "addUser": "system",
        "abstractDataId": "45b6f1ad43bb44db",
        "title": "记录温湿度属性",
        "deviceId": "970bdf41-7818-4718-b00e-eb14a08b6372",
        "deviceName": "温湿度传感器1751939165",
        "deviceStatus": 0,
        "deviceSecret": "9d4e0d3949134bd5",
        "abstractIdentifier": "Record",
        "payload": "temperature&-8.41,humidity&14.62,batteryLevel&90",
        "location": "0 0 0"
    }
}
],
"interfacePath": "/bv-microservice-digitaltwin-center/sceneLinkage/bvSpaceAbstractTemplate/findSpaceAbstractResultSet",
"requestType": 1,
"intelligentSpacesTree": {
    "bf029ce259a74835": {
        "intelligentSpacesId": "bf029ce259a74835",
        "intelligentSpacesParentId": null,
        "intelligentSpacesParentName": null,
        "intelligentSpacesName": "温湿度传感器",
        "intelligentSpacesCategory": 0,
        "intelligentSpacesType": 2,
        "intelligentSpacesJointId": null,
        "intelligentSpacesCreateMode": 0
    }
}

```

intelligentSpacesCreateMode": 0,
"intelligentSpacesReferenceId": null,
"intelligentSpacesDescription": null,
"intelligentSpacesSequence": null,
"subordinateIntelligentSpacesList": null,
"subordinateIntelligentSpacesCount": null,
"intelligentSpacesResourceList": null,
"intelligentSpacesResourceCount": null,
"hasStructIntelligentSpacesParent": null,
"isSubscribed": null
}
}
}
}

方式2：低代码设计器图表

DTS数据孪生架构提供快速接入查看DTS设备数据的平台功能[低代码设计器](#)，详细可视化功能可查看 [《BigView低代码设计器教程》](#)。

订阅管理概览

订阅管理

订阅管理允许应用服务以发布/订阅（Pub/Sub）模式获取实体的状态变更、属性上报或异常事件，由系统将指定消息流转至下游的订阅者。

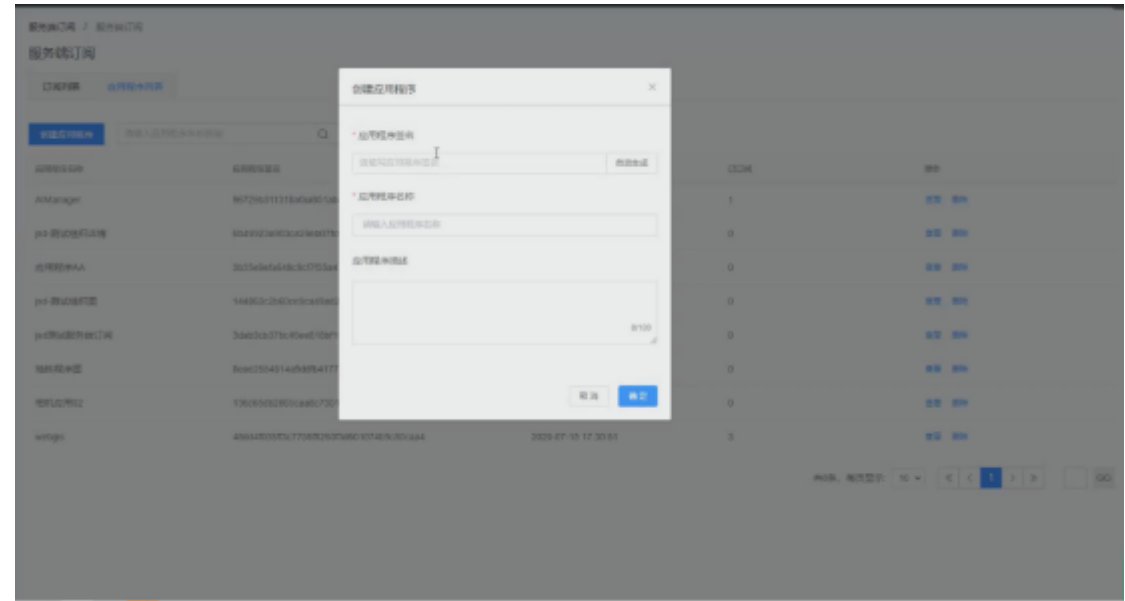
[!NOTE] 关于订阅机制与订阅者的基本概念，请参考[概念定义 - 订阅管理](#)。

创建订阅者

订阅者指需要向服务端进行订阅的服务端或应用平台。

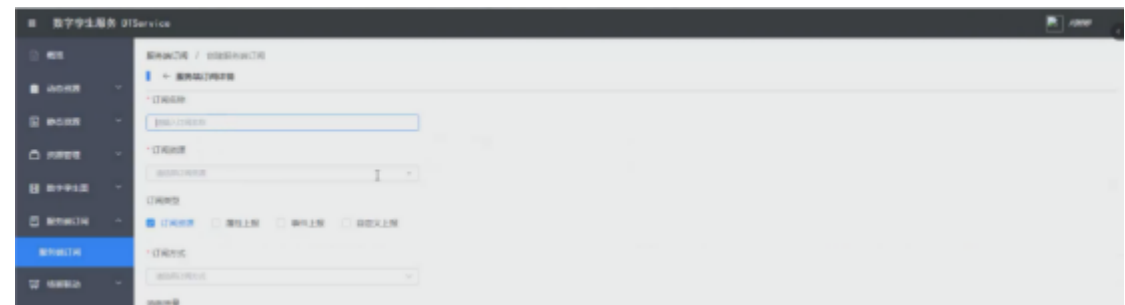
用于在订阅体系内构建与第三方平台对接的用户与角色组。

[!NOTE] 关于消费者扩展体系的具体定义，请参考[概念定义 - 消费者](#)。



创建订阅关系

订阅关系提供订阅 DTS 中所有资源消息并将其分发至第三方系统或应用的能力。





参数	说明
产品实体	<p>选择要订阅的消息源实体。可在产品实体页中选择全部实体、授权实体或未授权实体。</p> <p>说明： 实体配置订阅前必须先通过权限策略授权，未授权的实体需要先在页面申请授权。</p>
订阅类型	<p>选择订阅的数据通道类型：</p> <ul style="list-style-type: none">• 订阅资源： 获取底层资源的访问权限（这是订阅属性和事件的前提条件）；• 属性上报： 当实体属性发生变化时，转发属性消息；• 事件上报： 当实体产生事件时，转发事件消息；• 自定义上报： 当网关发生业务改变时发送自定义格式的消息。
订阅事件	<p>选择对应产品 Profile 中定义的具体事件名称。</p> <p>说明： 仅在“订阅类型”选择为“事件上报”时展示。</p>
订阅自定义Topic	<p>选择对应产品具有发布权限的自定义 Topic 路径。</p> <p>说明： 仅在“订阅类型”选择为“自定义上报”时展示。</p>
订阅方式	<p>目前仅支持使用 Redis 队列分发。</p>
消息质量	<p>上行数据流转的服务质量等级（QoS）：</p> <ul style="list-style-type: none">• 默认：代表 <code>QoS=0</code>• 二级：代表 <code>QoS=2</code> （目前暂不支持 <code>QoS=1</code>）
订阅者	<p>选择接收消息的第三方服务端或订阅者（支持多选）。</p> <p>说明： 点击“选择订阅者”可创建并授权消费者应用，每个应用会获得独立的 <code>appSecret</code> 与 <code>consumerSecret</code>，并匹配生成对应的订阅 Topic（例如：<code>/serverSubscription/{appKey}/{consumerSecret}/event/sub</code>）。</p> <p>创建应用程序（applicationProgramSecret） 应用程序名称支持中文汉字、英文字母、数字和下划线（<code>_</code>），长度范围为4个~30个字符。一个汉字占两个字符。 应用程序签名必须与入库验证系统记录的平台签名一致或与应用程序统一协商后生成录入。 应用程序备注。 系统将自动生成 应用程序Topic 事件 <code>/serverSubscription/\${applicationProgramSecret}/\${consumerSecret}/event/sub</code> 属性 <code>/serverSubscription/\${applicationProgramSecret}/\${consumerSecret}/property/sub</code> <code>consumerSecret</code>为消费者，每个平台下的用户体系，用来区分设备权限形成消息级别多级管控； 创建消费者（consumerSecret）</p>

参数	消费者名称 (Consumer Name) 消费者建立在应用程序的基础上，一个应用程序对应多个消费者； 消费者名称支持中文汉字、英文字母、数字和下划线（_），长度范围为4个~30个字符。一个汉字占两个字符。 消费者签名全局唯一，可自主录入或者自动生成，数字孪生平台提供自动生成功能。 消费者备注。

操作	订阅关系的流转动作，目前支持： • 消息缓存 ：将消息在 Redis 中进行缓存（保留 3×24 小时，以 messageID 为 key），用于临时存储、模板渲染及历史追溯。 若事件配置了“输出模板”，系统将默认勾选此项。
----	---

订阅关系标签

在创建或编辑订阅关系时，可以设置标签用于对订阅关系进行分类与筛选。该功能支持“可选择 + 可输入”的组合模式与两级标签管理，既可以从下拉列表中快速选择已有的分组或标签，也允许自定义输入新标签。

参数	说明
标签分组 (一级标签)	标签的类别或分组名称，支持快速选择已有分组或自定义输入创建（如 SubscriptionType 、 Project ）。
标签 (二级标签)	具体的标签值，支持快速选择已有标签或自定义输入创建（如 AlarmPush 、 Phase-1 ）。

HTTP回调推送

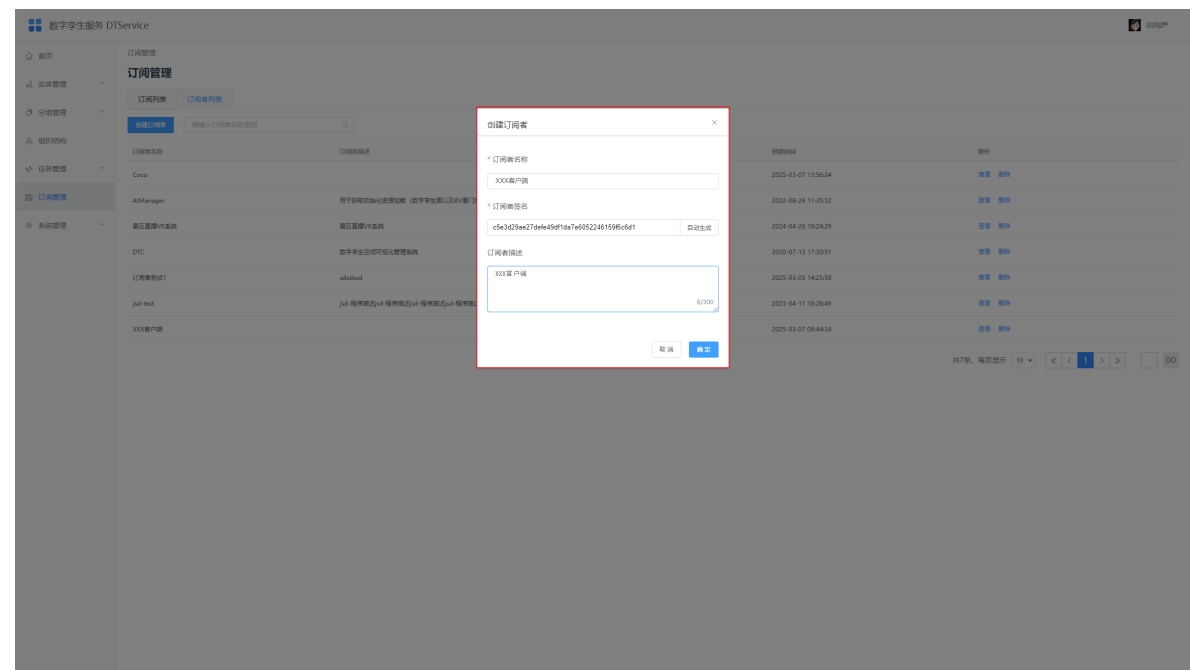
HTTP回调推送

通过本文你可以了解到通过HTTP方式从DTS订阅实体状态消息的流程。

示例

创建订阅者

进入订阅管理，选择订阅者列表。为服务端或订阅者创建订阅者身份。创建一个名为“XXX客户端”的订阅者，选择自动生成签名。



设定回调地址

进入“XXX客户端”编辑，选择“回调地址”一栏。添加一个Http的回调地址，命名为“事件回调”，输入接收Post请求的Http服务端地址“<http://192.168.100.236:4523/m1/4795394-4449583-default/receivedtsdata/event>”。



* 推送地址

http://192.168.100.236:4523/m1/4795394-4449583-default/receivedtsd

确定

取消

订阅资源

为“XXX客户端”添加订阅资源。进入订阅列表添加一个新的订阅，设置订阅的资源，选择订阅“智能冰箱”和“智能插座”产品。订阅类型勾选事件上报。订阅方式设定为HTTP，

订阅者勾选“XXX客户端”，并配置回调地址为“事件回调”。完成配置后，选择确定。保存新建的订阅。

应用程序						✕
<div>请输入应用产品名称查询</div>				<div>全部</div>		已选
<div></div>	订阅者名称	签名	回调地址	创建时间	描述	
<div></div>	Coco	420103faa9cbfd1d...	未配置	2025-03-07 13:56:24		
<div></div>	AIManager	96729b311318afa...	未配置	2022-08-26 11:35:52	用于获取初始化资...	
<div></div>	景区直播VR系统	59149b5cb7260b8...	未配置	2024-04-26 16:24:29	景区直播VR系统	

☐	DTC	48dd4f008ff3c775...	未配置	2020-07-15 17:30:51	数字孪生空间可视...
☐	订阅者测试1	ac86f98ab559857...	未配置	2025-03-05 14:25:58	sdsdssd
☐	jsd-test	fbcd4466364aa8dd...	未配置	2023-04-11 16:28:49	jsd-程序描述jsd-程...
☒	XXX客户端	c5e3d29ae27defe...	http://192.168.100.23t	2025-03-07 09:44:34	
☐	ryf-test	e9c43ba0c6f5471...	未配置	2025-03-10 16:02:46	http协议事件订阅

共8条，每页显示

10 ▾

«

<

1

>

»

GO

智能冰箱、智能插座发生事件变化时，对应Http服务端查看到实体数据。

详细的数据格式可参考[订阅资源流转-协议格式](#)。

RedisMQ订阅推送

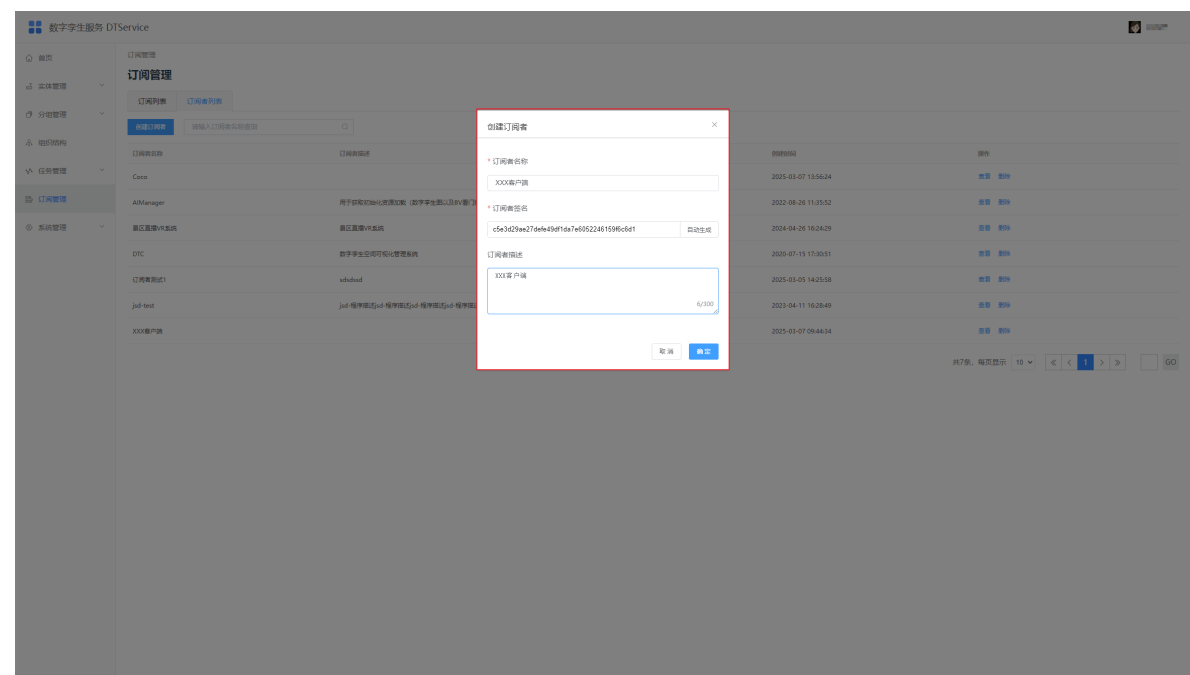
RedisMQ订阅推送

通过本文你可以了解到通过Redis方式从DTS订阅实体状态消息的流程。

示例

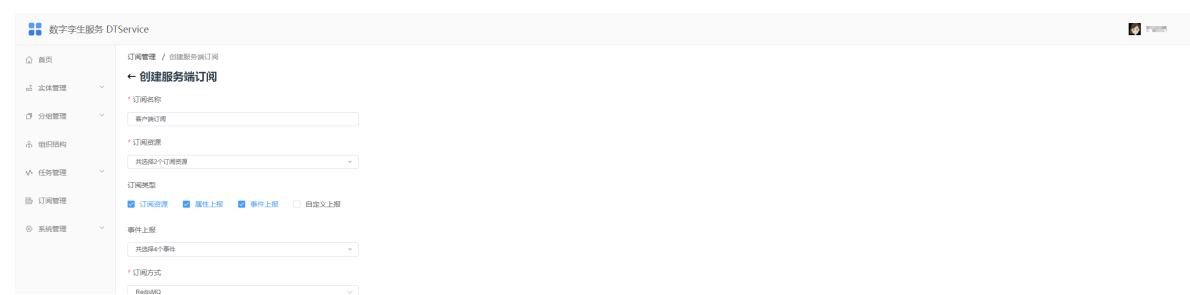
创建订阅者

进入订阅管理，选择订阅者列表。为服务端或订阅者创建订阅者身份。创建一个名为“XXX客户端”的订阅者，选择自动生成签名。



订阅资源

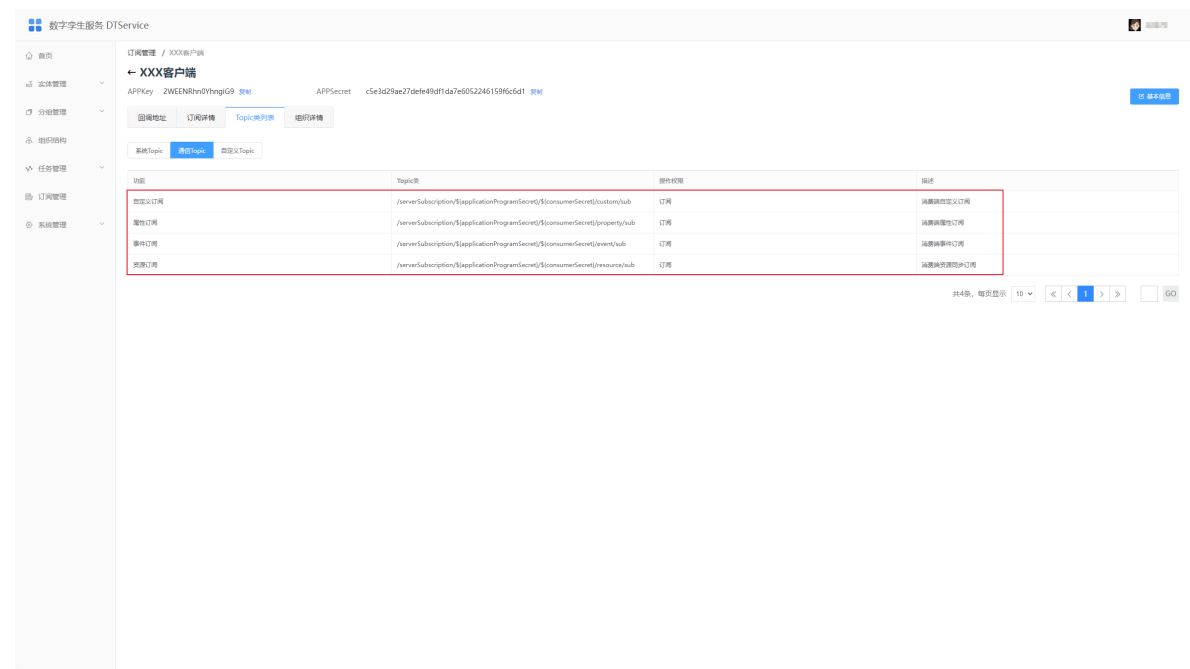
为“XXX客户端”添加订阅资源。进入订阅列表添加一个新的订阅，设置订阅的资源，选择订阅的产品。订阅类型勾选上订阅资源、属性上报、事件上报。订阅方式设定为RedisMQ，订阅者选择“XXX客户端”。





查询订阅通道

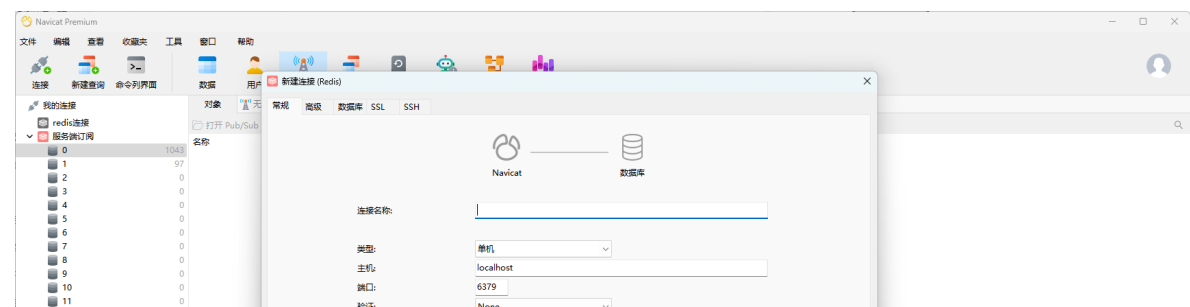
进入订阅管理，查看“XXX客户端”的详情。选择Topic通道下的通信Topic可查看使用Reids订阅的通道路径。

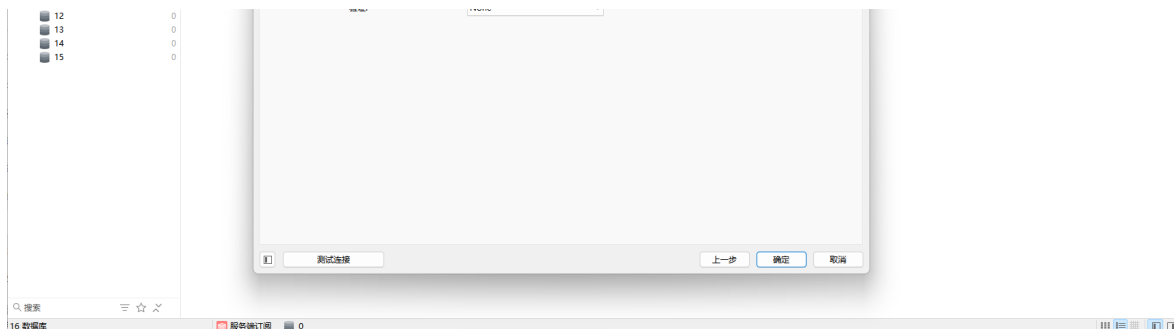


测试使用Redis订阅通道

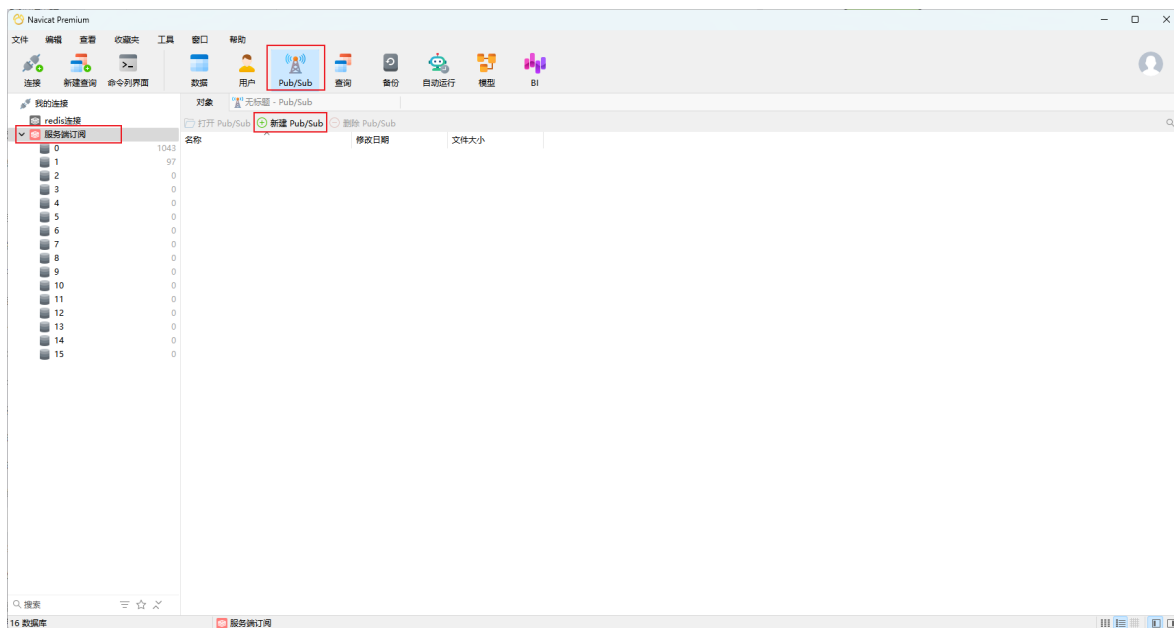
使用Navicat连接Redis服务器测试获取订阅数据内容。也可使用其他Redis工具或通过编写代码连接。订阅者接收到消息后可以
对数据自行进行逻辑处理。

在Navicat下新建Redis连接，输入Redis服务器地址，测试连接通过后点击确定。



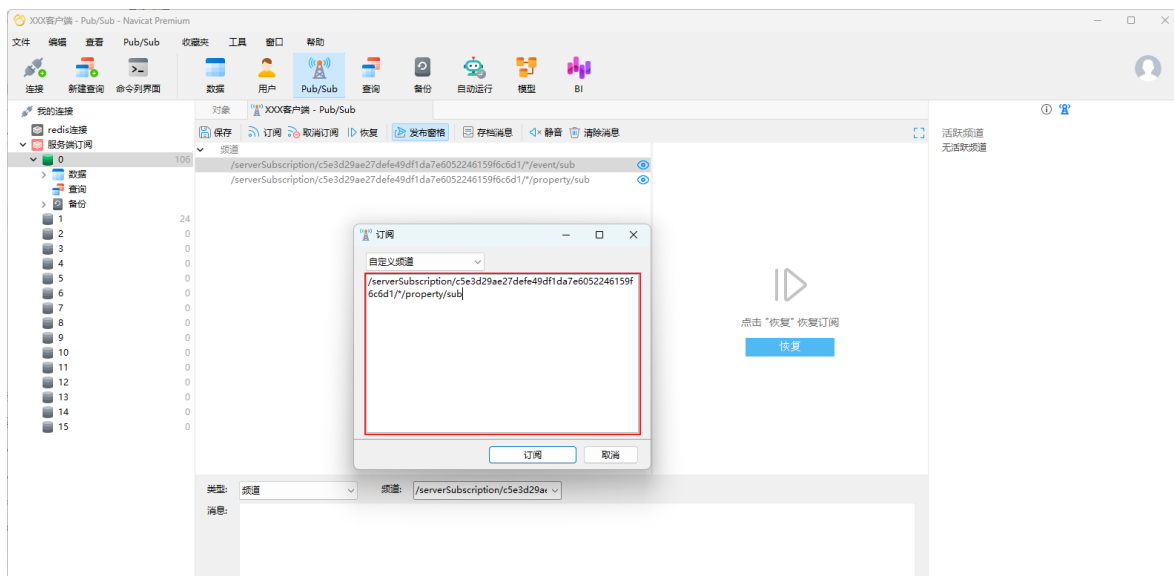


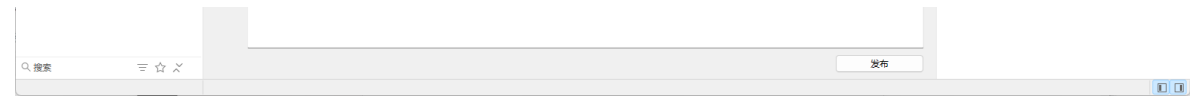
打开连接，选择Pub/Sub功能，新建一个Pub/sub对话。



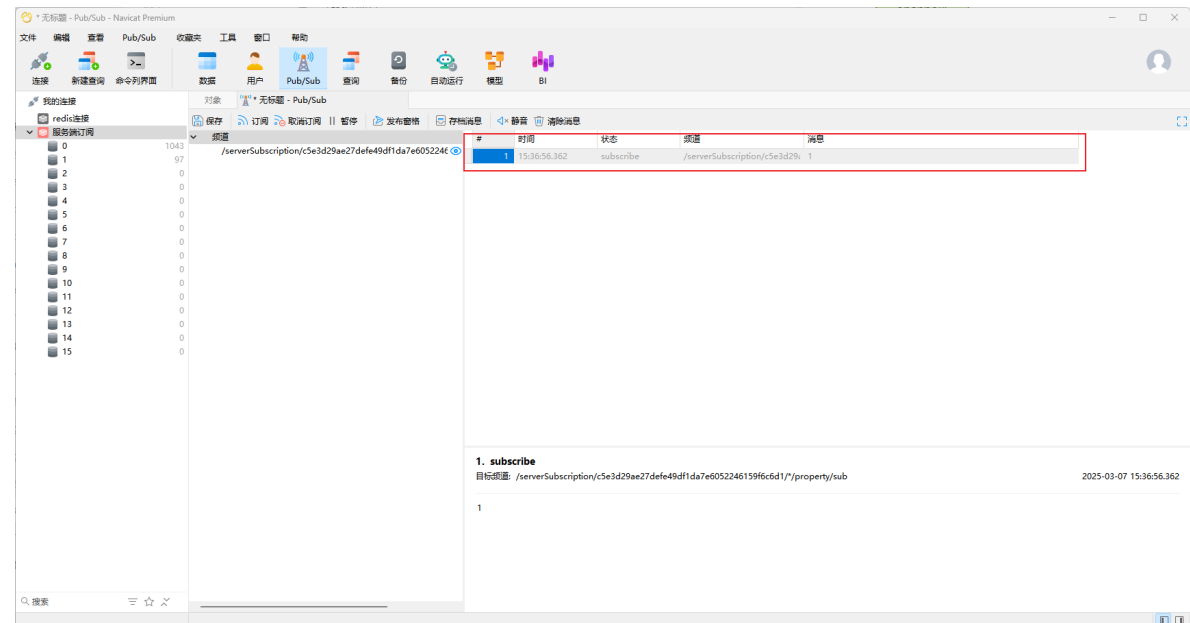
选择订阅，输入需要订阅的通道。从通信Topic下复制出属性订阅的通道号/serverSubscription/\${appSecret}/\${consumerSecret}/property/sub，注意：需要将其中 `appSecret` 替换为订阅者的身份签名 `APPSECRET`。

若未创建消费者，需 `${consumerSecret}` 使用通配符 `*` 替代。

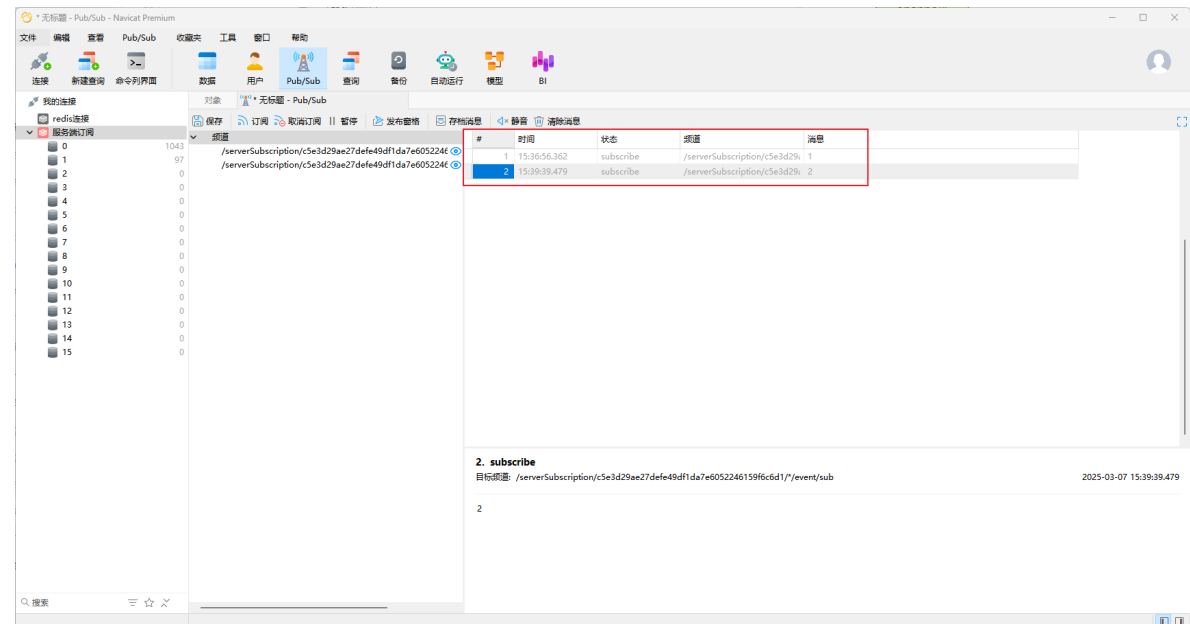




输入完成后，点击订阅。



同样的方式创建接收事件订阅的订阅通道/serverSubscription/\${appSecret}/\${consumerSecret}/event/sub。



智能冰箱、智能插座发生属性、事件变化时，就可通过订阅的通道查看到实体数据。

详细的数据格式可参考[订阅资源流转-协议格式](#)。



4. message

目标频道: /serverSubscription/c5e3d29ae27defe49df1da7e6052246159f6c6d1/* /event/sub

2025-03-10 14:19:13.720

```

{"abstractContent": "设备173926000位置发生实时温度值更新通知事件上报, 请及时查看", "callbackURL": "http://192.168.7.37:48087/bv-microservice-server-subscription-transpond/callback?type=2", "description": "设备173926000: 760a1d89-5f16-4aa3-bd2a-802e45b4f2b5", "deviceName": "设备1739260500", "deviceSecret": "4f2af114d34d4bfc", "deviceStatus": "1", "identifier": "TemperatureUpdate", "location": "0 0 0", "messageId": "e560298e2ac84c78", "name": "实时温度值更新通知", "payload": {"currentTemperature": 7, "unit": "C", "compartment": "Refrigerator", "previousTemperature": 18}, "productId": "OWqgR0cJmLw8XJ", "qos": "0", "templateId": "f5c9fd9d2624fcec", "templateURL": "http://192.168.100.22:7521/preview?code=component_ST9ENJAoet&messageId=e560298e2ac84c78&BV_WIDTH=450&BV_HEIGHT=320", "timestamp": "174158755368", "urlName": "http://192.168.100.22:7521/preview?code=component_ST9ENJAoet&messageId=e560298e2ac84c78&BV_WIDTH=450&BV_HEIGHT=320"}

```


WebSocket订阅推送

WebSocket 订阅推送

本文面向 GIS 平台或第三方订阅者客户端，说明如何通过 WebSocket 接收平台推送的订阅消息。

1. 接入前提

第三方客户端能连接 WebSocket 前，需先从平台方获取以下信息：

- 1. DTS已为该订阅者开通 WebSocket 订阅方式。
- 2. DTS提供 appSecret 。
- 3. 如按消费者隔离接收，DTS提供 consumerSecret ；没有消费者时使用 default 。
- 4. DTS提供可访问的 WebSocket Server 地址。

2. WebSocket Server 从哪里找

WebSocket Server 地址由DTS服务方提供。第三方客户端只需要使用DTS服务方给出的域名、IP 或测试地址进行连接。

本地或联调环境示例：

```
ws://{host}:48091/serverSubscription/{appSecret}/{consumerSecret}
```

HTTPS/WSS 环境示例：

```
wss://{domain}/serverSubscription/{appSecret}/{consumerSecret}
```

其中：

参数	来源	说明
appSecret	DTS服务方提供	订阅者密钥 (appSecret)
consumerSecret	DTS服务方提供	消费者密钥
default	DTS服务方约定	没有消费者时使用

没有消费者时，地址写成：

```
ws://{host}:48091/serverSubscription/{appSecret}/default
```

注意： /serverSubscription/{appSecret}/{consumerSecret} 是 WebSocket 握手地址，不是普通 HTTP 接口。直接用浏览器或 HTTP 工具访问会返回 400 ，提示使用 WebSocket 连接，这是预期行为。

3. 连接参数从哪里获取

WebSocket 通道格式是固定的，客户端不需要选择不同通道。需要从DTS服务侧获取的是通道中的 `appSecret`，以及按消费者隔离时的 `consumerSecret`。

可在 BV DTS 数字孪生服务中查看 WebSocket 通用通道，路径为：

BV DTS 数字孪生服务 -> 订阅管理 -> 订阅者列表 -> Topic类别表 -> WebSocket通用

WebSocket 通用通道格式为：

`/serverSubscription/{appSecret}/{consumerSecret}`

其中：

参数	获取方式	说明
<code>appSecret</code>	从订阅者对应的订阅者信息或 <code>WebSocket通用</code> 通道中获取	订阅者密钥 (appSecret)
<code>consumerSecret</code>	从订阅者对应的组织详情中获取， 操作路径为：创建组织 -> 创建消费者 -> 消费者绑定资源	消费者密钥
<code>default</code>	没有消费者时使用	默认消费者占位值

客户端连接时把占位符替换成真实密钥即可。属性、事件、资源、自定义消息不再使用不同 WebSocket 地址区分，而是在收到消息后通过 `subscriptionType` 和业务字段判断。

4. 连接成功与失败

连接成功后，服务端会返回：

```
{
  "type": "connected",
  "code": 0,
  "message": "success"
}
```

鉴权失败时，WebSocket 会被服务端关闭，关闭码为 `1008`，含义是策略校验失败。常见原因：

- 地址中的 `appSecret` 或 `consumerSecret` 不正确。
- DTS服务方尚未为该应用开通 WebSocket 订阅。
- 订阅没有关联该订阅者或消费者。
- WebSocket Server 暂不可用。

5. 客户端需要发送的消息

5.1 心跳

客户端建议每 20 到 30 秒发送一次心跳：

```
{
  "type": "heartbeat"
}
```

也可以发送：

```
{
  "type": "ping"
}
```

服务端响应：

```
{
  "type": "pong",
  "code": 0,
  "message": "success"
}
```

心跳的主要作用是告诉DTS服务方连接仍然存活，并维持集群环境下的消息投递。生产客户端应持续定时发送心跳，不建议只在连接建立时发送一次。

建议客户端连续 2 到 3 次没有收到 `pong` 时主动重连。

5.2 QoS2 ACK

当收到 `qos=2` 的消息时，客户端处理完成后需要发送 ACK：

```
{
  "type": "ack",
  "messageId": "00f61d2f573241b695bf8e967636f0d3"
}
```

服务端响应：

```
{
  "type": "ack",
  "code": 0,
  "message": "success",
  "messageId": "00f61d2f573241b695bf8e967636f0d3"
}
```

如果 `messageId` 不属于待确认的 QoS2 消息，例如 QoS0 消息、已经 ACK 过的消息或已过期的消息，服务端会返回非 0 响应：

```
{
  "type": "ack",
  "code": 4004,
  "message": "message not pending or already acknowledged",
  "messageId": "00f61d2f573241b695bf8e967636f0d3"
}
```

QoS 策略：

QoS	客户端要求	说明
0	不需要 ACK	遥测默认策略，在线即推，离线不补偿

QoS	暂时不需要完整语义	预留保留字段
2	需要 ACK	未 ACK 时平台可能重试，客户端需要按 <code>messageId</code> 做幂等处理

6. 服务端下发消息格式

WebSocket 下发消息体与 HTTP、RedisMQ 的业务消息体保持一致，不额外增加外层 `payload` 信封。平台仅额外保留 `subscriptionType`，用于客户端判断当前消息属于资源、属性、事件还是自定义 Topic。

```
{
  "messageId": "00f61d2f573241b695bf8e967636f0d3",
  "productId": "OWqrR0cJzMLvx8DJ",
  "productName": "智能冰箱",
  "deviceId": "283d7823-1d72-4081-b6e8-9fecf59dfc43",
  "deviceName": "123456",
  "deviceGuid": "283d7823-1d72-4081-b6e8-9fecf59dfc43",
  "qos": 0,
  "timestamp": 1780920540168,
  "subscriptionType": 1,
  "payload": {
    "temperature": -19,
    "status": 1
  }
}
```

字段说明：

字段	说明	第三方是否通常需要
<code>messageId</code>	消息编号，QoS2 ACK 使用	是
<code>productId</code>	产品编号	是
<code>deviceId/deviceGuid</code>	设备编号	是
<code>subscriptionType</code>	订阅消息类型	是
<code>qos</code>	消息质量等级	是
<code>timestamp</code>	消息时间戳，毫秒	是
<code>payload</code>	属性、事件或自定义消息的业务内容	是

7. 消息类型怎么判断

优先使用外层 `subscriptionType` 判断：

<code>subscriptionType</code>	类型	说明
0	资源同步	产品、属性、事件信息资源同步

0 subscriptionType	资源同步 类型	产品、头体、学生体等资源变更 说明
1	属性上报	设备属性或遥测数据
2	事件上报	告警、事件、任务状态等
3	自定义 Topic	第三方自定义消息

注意：业务消息体中也可能存在名为 `type` 的字段，例如事件类型 `0 信息 / 1 告警`。客户端不要把业务字段 `type` 当成 WebSocket 消息类型。

可辅助判断的业务字段：

类型	常见字段
属性上报	<code>payload</code> 中是属性键值，如温度、电量、经纬度等
事件上报	<code>identifier</code> 、 <code>name</code> 、 <code>description</code> 、 <code>type</code>
资源同步	<code>resourceType</code> 、 <code>syncMethod</code> 、 <code>subscribeChangeList</code> 、 <code>twinsList</code>
自定义 Topic	<code>customTopic</code> 或自定义消息体

8. 推荐客户端处理流程

1. 从DTS服务方获取 WebSocket Server 地址、`appSecret` 和 `consumerSecret`。
2. 确认DTS服务方已为该应用开通 WebSocket 订阅。
3. 连接 `ws://{host}:48091/serverSubscription/{appSecret}/{consumerSecret}`。
4. 收到 `connected` 后启动定时心跳，每 20 到 30 秒发送一次。
5. 收到业务消息后先读取外层 `subscriptionType/qos/messageId`。
6. 根据 `subscriptionType` 分发到属性、事件、资源或自定义消息处理逻辑。
7. 业务处理读取 `payload`。
8. 如果 `qos=2`，处理成功后发送 ACK。
9. 连接断开或连续多次无 `pong` 时重连。

9. JavaScript 测试示例

```
const appSecret = "420103faa9cbfd1d267d99049a7fed5bb83e9774";
const consumerSecret = "default";
const ws = new WebSocket(`ws://127.0.0.1:48091/serverSubscription/${appSecret}/${consumerSecret}`);

let heartbeatTimer;

ws.onopen = () => {
  heartbeatTimer = setInterval(() => {
    ws.send(JSON.stringify({ type: "heartbeat" }));
  }, 25000);
};

ws.onmessage = (event) => {
  const message = JSON.parse(event.data);

  if (message.type === "connected" || message.type === "pong" || message.type === "ack") {
    console.log("control message:", message);
    return;
  }
}
```

```
}

const { subscriptionType, qos, messageId } = message;

switch (subscriptionType) {
  case 0:
    console.log("resource sync:", message);
    break;
  case 1: {
    console.log("property:", message);
    const propertyValues = message.payload || {};
    console.log("property values:", propertyValues);
    break;
  }
  case 2:
    console.log("event:", message);
    break;
  case 3:
    console.log("custom topic:", message);
    break;
  default:
    console.log("unknown message:", message);
}

if (qos === 2 && messageId) {
  ws.send(JSON.stringify({ type: "ack", messageId }));
}
};

ws.onclose = (event) => {
  clearInterval(heartbeatTimer);
  console.log("websocket closed:", event.code, event.reason);
};

ws.onerror = (error) => {
  console.error("websocket error:", error);
};
```

10. 联调检查清单

1. 客户端能成功建立 WebSocket 连接。
2. 连接后能收到 `connected`。
3. 客户端发送心跳后能收到 `pong`。
4. 属性上报时能收到 `subscriptionType=1`。
5. 事件上报时能收到 `subscriptionType=2`。
6. 如收到 `qos=2` 消息，客户端能按 `messageId` 发送 ACK。
7. 主动断开或网络异常后，客户端能自动重连并继续接收消息。

系统管理

系统管理

导入导出

导入导出

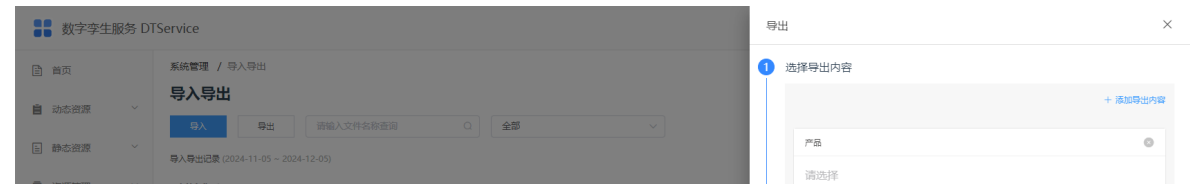
导出

将平台中的数据导出为成为Json文件，方便项目数据的移植。目前支持可导出数据为产品、网关、组织结构、场景、数据推送函数、时序数据查询接口等内容。可根据需求选择导出的内容。

名称	可导出内容
产品	<code>ProductKey</code> 、 <code>ProductSecret</code> 、产品基础信息、产品标签、Topic 类列表、产品 Profile（包含属性、事件与服务的定义，暂不包含事件模板）以及消息解析脚本。 说明： 产品导出时 不包含 其下的实体列表。
网关	<code>GatewayKey</code> 、 <code>GatewaySecret</code> 、网关配置信息（含网关与产品的关联关系）、网关标签、网关属性以及发现实体接口列表。 说明： 网关导出时 不包含 其下挂载的实体列表。
组织结构	仅包含属于 DTS 的孪生体组织结构（即“DTS内部结构”下的所有节点）。 注意： 当前导出功能正在优化中。
数据推送函数	函数基础信息、JavaScript 函数脚本。
任务	任务基本信息、任务规则、任务与数据推送函数以及产品实体之间的关联关系。
时序数据查询接口	接口配置信息（如查询字段与参数设置）及接口内容。 说明： 任务与时序接口导出时，会自动带入并保留它们与对应数据推送函数之间的关联关系。

创建导出

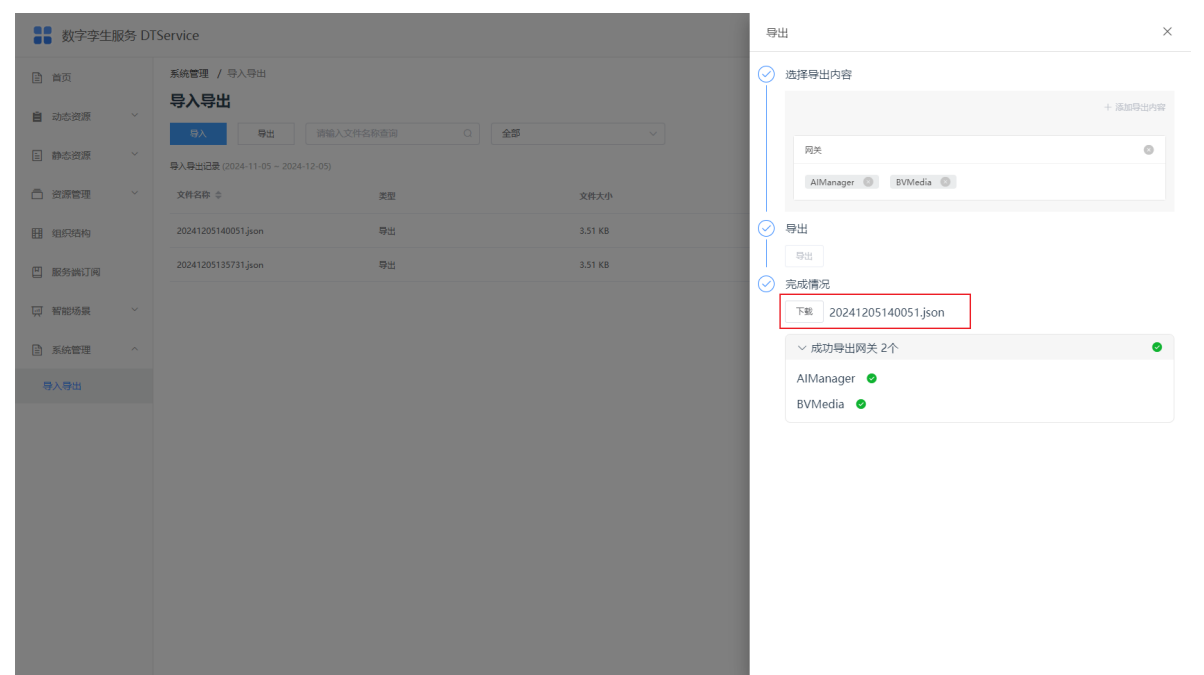
根据步骤引导，按步骤完成导出设置。首先设置需导出的内容，然后开始导出。





下载文件

导出完成后，可查看导出内容的明细。如有导出失败的内容，可尝试再次导出。导出后，即可下载文件到本地电脑以备所需。



导入

将导出的平台文件，导入到目的平台中。由于实体均不支持导出，所以导入的任务、数据推送函数等内容中，可能存在实体缺失的设置。这一部分缺失，需导入后，根据新获取的实体列表进行手动修改。或由于其他原因引起的数据缺失，需要手动修改。





创建导入

根据步骤引导，按步骤完成导入设置。上传导入的数据文件，然后开始导入。

完成导入

导入完成后，可查看导入内容的明细。如有导入失败的内容，可尝试再次导入。导入完成后，建议谨慎检查数据是否正确导入，存在错误请及时修改。

存储与删除

在导入导出列表中，删除导入导出记录并同时删除导入或导出的文件。平台将导入或导出的文件默认存储30天。30天后将自动清除导入导出记录和文件。

日志服务

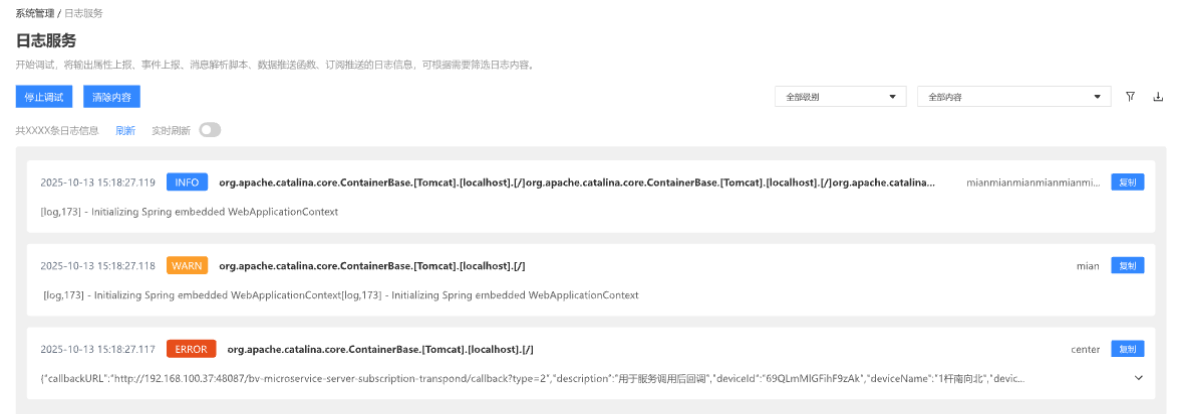
日志服务

功能概述

日志服务是系统管理模块中的重要组成部分，主要用于记录和展示系统运行过程中的各类日志信息。通过日志服务，用户可以追踪设备数据交互的全过程，便于问题排查和系统监控。

主要功能

- **实时日志监控**：支持实时刷新查看最新日志
- **日志类型筛选**：可按不同业务类型筛选日志
- **日志级别过滤**：支持按INFO、WARN、ERROR等级别过滤
- **调试模式**：支持开启/停止调试，记录关键操作日志



操作说明

进入日志服务

1. 登录系统后，点击左侧导航菜单的 **系统管理**
2. 在子菜单中找到并点击 **日志服务**，进入日志管理页面

调试控制

开始调试

- 点击 [**开始调试**] 按钮，系统将开始记录以下类型的日志信息：
- 属性上报
- 事件上报
- 消息解析脚本输出
- 数据推送函数输出
- 订阅推送信息

停止调试

- 点击 [**停止调试**] 按钮，系统将暂停日志记录

清除内容

- 点击 [**清除内容**] 按钮，可清空当前显示的日志列表

日志查看

刷新日志

- **手动刷新**：点击 **刷新** 按钮，获取最新日志
- **实时刷新**：开启 **实时刷新** 开关，系统将自动滚动显示最新日志

日志计数

- 页面顶部显示当前日志总数（例如：共 XXXX 条日志信息）

日志筛选

按级别筛选

点击 **全部级别** 下拉菜单，可选择：

- 全部级别
- TRACE
- DEBUG
- INFO
- WARN
- ERROR

按内容类型筛选

点击 **全部内容** 下拉菜单，可选择以下日志类型：

筛选选项	说明
全部内容	显示所有类型的日志
实体注册	实体注册上报到系统的原始报文、日志
实体映射（属性）	实体属性上报到系统的原始报文、日志
实体映射（事件）	实体事件上报到系统的原始报文、日志
实体反控（服务）	数据经过消息解析的输出，包括脚本内编写的控制台输出
数据推送函数	数据经过数据推送函数的输出，包括脚本内编写的控制台输出
订阅推送	推送订阅的消息到消息接收器

日志内容解读

日志格式示例

日志条目通常包含以下信息：

- 时间戳：如 2025-10-13 15:18:27.119
- 日志级别：INFO / WARN / ERROR
- 日志来源：如 org.apache.catalina.core.ContainerBase
- 日志内容：具体的日志消息或数据报文

常见日志类型示例

系统日志示例

2025-10-13 15:18:27.119 INFO org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/] - Initializing Spring embedded WebApplicationContext

数据推送日志示例

2025-10-13 15:18:27.117 ERROR org.apache.catalina.core.ContainerBase.[Tomcat].[localhost].[/]
["callbackURL":"http://192.168.100.37:48087/bv-microservice-server-subscription-transpond/callback?type=2","description":"用于服务调用后回调","deviceId":"69QLn"]

使用场景建议

场景	推荐操作
----	------

场景	推荐操作
设备接入调试	开启调试，筛选"实体注册"和"实体映射"日志
脚本开发测试	开启调试，关注"实体反控（服务）"中的控制台输出
数据推送异常排查	筛选"数据推送函数"和"订阅推送"，开启实时刷新
系统启动问题分析	关闭筛选，查看全部级别的系统日志

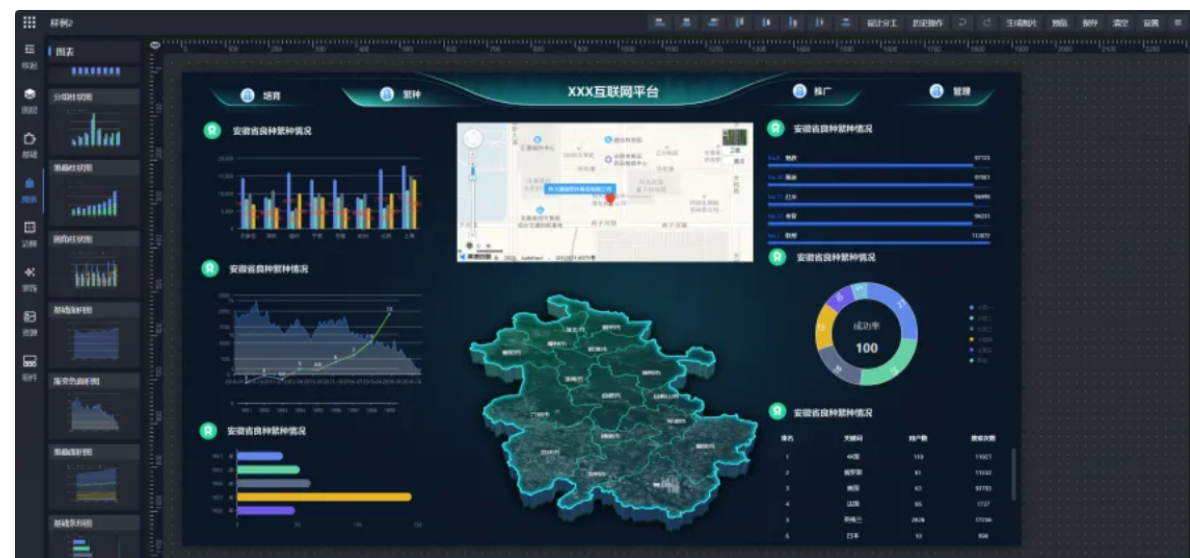
注意事项

- 1. **日志量控制**：长时间开启调试可能产生大量日志，建议按需开启
- 2. **实时刷新影响**：开启实时刷新可能会影响页面性能，日志量较大时可关闭此功能
- 3. **日志保留**：系统对日志保留时间有限制，重要日志请及时导出或记录

低代码设计器

低代码设计器

DTS数据孪生的关联产品，DTS的网关、实体状态编辑中嵌入其模板使用，除了支持市面常规的数据库、数据集接入外，在数据集中特别增加了DTS产品、实体的数据源。基于SpringBoot、MyBatisPlus、ElementUI、G2Plot、Echarts等技术栈的低代码设计器，具备布局分组管理、看板设计、预览能力，支持MySQL、Oracle、PostgreSQL、JSON、DTS实体等数据集接入，对于复杂数据处理还可以使用Groovy脚本数据集，使用简单，易上手。



详细操作指南请参考[BigView低代码设计器教程](#)

开发者参考手册

开发者参考手册

DTS（数字孪生服务）提供了丰富的开发者工具与对接接口。本手册提供了外部系统对接、消息解析脚本以及内置 API 的大纲指引（详细内容已迁移至独立的《[DTS对接开发指南](#)》），同时保留了空间拓扑与 ImageMask 分析、AI 算法快速落地以及 BV 相机 Profile 介绍的完整内容。如需查阅这些完整本地章节，请点击下方对应链接。

1. 外部系统对接开发

DTS 提供了标准化的数据分发服务，支持第三方系统通过以下方式进行对接开发：- **数据订阅**：支持通过 Redis 消息队列、WebSocket 长连接或 HTTP 回调接收设备上报的 BV JSON 数据。- **服务反控**：第三方系统携带 APIKey 调用标准 API，实现对实体的服务调用与控制。- **详细参考**：关于接口地址、请求/返回参数及鉴权方式，请前往 [DTS对接开发指南 - 外部系统对接](#) 查阅。

2. 消息解析脚本开发规范

对于采用自定义/透传数据格式的产品，DTS 允许在沙箱环境中编写脚本进行数据转换：- **脚本支持**：支持使用 JavaScript (ES6) 和 Python 语言编写消息解析脚本。- **上行解析**：将设备上报的透传/自定义格式数据转换为 DTS 标准的 BV JSON 格式。- **下行解析**：将 DTS 下发的标准 BV JSON 指令转换为设备可识别的特定指令格式。- **详细参考**：关于解析脚本的入参出参定义及编写规范，请前往 [DTS对接开发指南 - 消息解析脚本开发规范](#) 查阅。

3. 基础工具类 API 参考

在编写消息解析脚本、请求脚本或数据推送函数时，DTS 提供了多个内置 of 工具类：- **Console 类**：用于在沙箱环境中打印调试日志。- **CoreApi 类**：提供对实体、属性、事件及节点关系的查询与操作接口。- **FileStorage 类**：用于临时或持久化存储脚本执行中的文件。- **MockXMLHttpRequest 类**：在解析脚本中模拟 HTTP 请求。- **JtsGeometry 类**：提供空间拓扑计算与几何对象分析（如判断点是否在多边形内）。- **详细参考**：关于各工具类的 API 函数签名及使用示例，请前往 [DTS对接开发指南 - 基础工具类API参考](#) 查阅。

4. 空间拓扑与 ImageMask 高级分析

DTS 支持高级的空间地理计算与影像遮罩（ImageMask）分析：- **空间计算**：基于 JTS 提供相交、包含、距离计算等空间拓扑关系判定。- **ImageMask 遮罩**：支持利用影像遮罩技术进行地理围栏判定或特定区域的状态计算。- **详细参考**：关于空间数据格式、ImageMask 的算法逻辑及配置示例，请查看本地文档 [04.02_空间拓扑与ImageMask高级分析.md](#)。

5. AI 算法的快速应用落地

DTS 支持将外部 AI 算法服务化并快速应用于任务规则引擎中： - **算法注册**：在产品服务定义中注册外部 AI 算法的请求地址与入参。 - **任务联动**：在任务管理中，通过规则引擎配置在满足条件时自动调用该 AI 算法服务，实现智能闭环控制。 - **详细参考**：关于算法服务的接入规范、输入输出设计及落地案例，请查看本地文档 [04.03_AI算法的快速应用落地.md](#)。

6. 附录：BV 相机产品 Profile 介绍

介绍了大视景（BV）标准相机的产品本体描述（Profile）： - **功能定义**：包含相机常用的属性（如在线状态、云台角度）、事件（如人脸识别、移动侦测）以及服务（如云台控制、抓拍）。 - **详细参考**：关于 BV 相机 Profile 的完整 JSON 结构及字段定义，请查看本地文档 [04.04_附录：BV相机产品Profile介绍.md](#)。

空间拓扑与ImageMask高级分析

ImageMask的使用介绍

ImageMask是DTS中新增的一种特殊数据类型。其核心功能是通过空间过滤提升AI分析的精度。支持用户在图片上通过矢量绘制任意形状的掩膜区域，对AI分析服务的分析结果做二次的过滤使用，筛选出在自己关注区域内的分析结果，提高分析结果的准确度。内置WGS84地理坐标与像素坐标双向转换模块，支持遥感影像等地理空间数据的掩膜分析场景，适用于自动驾驶路径规划、卫星图像分析等垂直领域。

创建

- 1. 选择目标产品，点击编辑进入产品信息详情页面->选择产品Profile Tab页面。
- 2. 点击编辑草稿，选择添加一个属性。
- 3. 输入属性名称和标识符“area”，选择数据类型为**ImageMask（图片掩膜）**
- 4. 完成后点击退出编辑。
- 5. 在产品Profile所有编辑完成后，需点击**发布**。发布后的产品Profile才能被DTS识别。

添加功能

×

* 功能类型

属性

事件

服务

* 功能名称

area

* 标识符

area

* 数据类型

ImageMask(图片掩膜)

* 读写类型

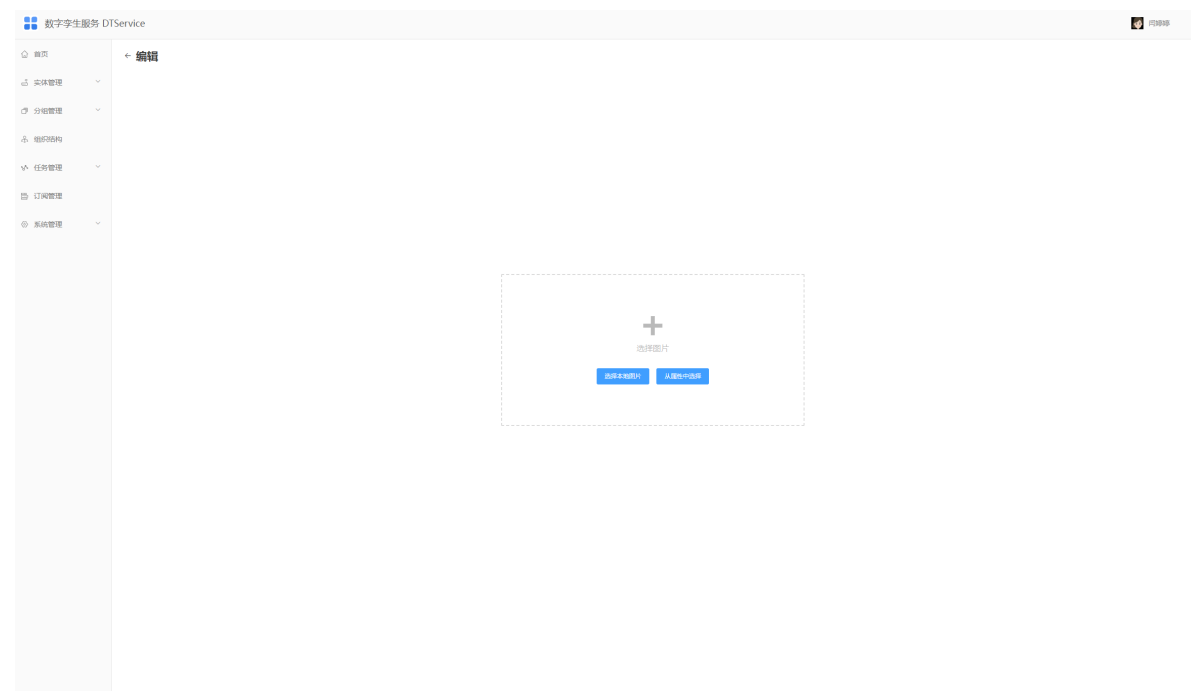
只读

读写

编辑

- 1. 选择目标产品下的实体。

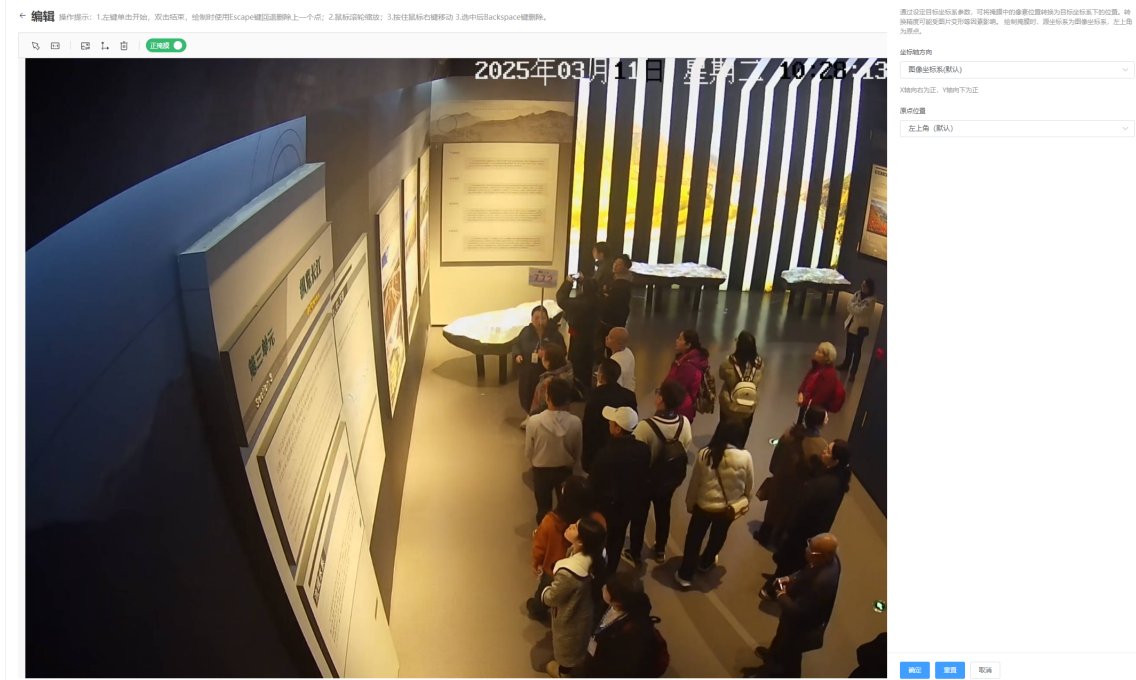
- 2. 点击编辑进入实体详情页。
- 3. 选择“实体状态”Tab页下的“属性值”标签页；
- 4. 选择“area”后的编辑功能，进入掩膜绘制页面。



- 1. 选择从本地电脑中上传一张图片或者从实体属性中选择某个属性值作为图片来源。
注意：默认不会保留临时选择的图片，【保存到属性】方可持久化存储，图片将更新至对应属性值。未保存的图片在退出后自动清除。
- 2. 设置图片来源后进入绘制；
- 3. 默认进入即为绘制状态，左键单击开始，双击结束，绘制时使用Escape键回退删除上一个点；
- 4. 在图片上绘制任意形状的掩膜区域，默认绘制为正掩膜，即只对此区域内的数据感兴趣。可修改为负掩膜，即排除此区域内的数据。



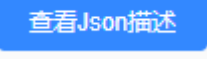
1. 默认绘制的坐标系为图像坐标系，坐标原点在左上角。如需将绘制的掩膜转换到自定义图像坐标系或地理坐标系下，可设置目标坐标系的参数。用于之后的脚本中进行坐标转换。



2. 绘制完成，点击保存。

功能操作提示：

- 1. 鼠标滚轮缩放；
- 2. 按住鼠标右键移动；
- 3. 选中 “拾取”功能，进入“拾取状态”。左键单击选中掩膜，使用Back scape键删除绘制的掩膜；左键单击选中顶点，鼠标拖拽修改顶点位置；

4. 取消选中  “拾取”功能，进入“绘制状态”。左键单击开始，双击结束，绘制时使用Escape键回退删除上一个点。默认为“绘制状态”。
5. 选择  “适应图片到画布”功能，图片大小变为与默认画布大小一致。
6. 选择  “切换图片”功能，可修改背景图片的来源，重置图片会重置所有掩膜区域和目标坐标系的参数。
7. 选择  “目标坐标系”功能，可设置目标坐标系的参数。自定义原点，选择“绘制原点”后在图片上点击选择新的原点。在平面地图坐标系下，可采用地理坐标，通过输入原点坐标值和设置空间分辨率确定后续转换地理坐标的所需参数。设置空间分辨率，在图片上绘制一条线段。
8. 选择  “清空掩膜”功能，删除当前绘制的所有掩膜。
9. 选择  “查看Json描述”功能，可查看当前ImageMask的完整属性值内容；
10. 选择  “保存”功能，保存当前绘制的掩膜和目标坐标系参数；

使用

1. 选择目标产品，点击编辑进入产品信息详情页面->选择消息解析脚本 Tab页面。
2. 在实体映射（上行）脚本中，写入收到AI分析服务结果之后与ImageMask进行二次过滤的逻辑；
3. 脚本编写完成后，保存脚本。

作者：北京-产品-yantings 创建时间：2025-05-28 14:08
最后编辑：北京-产品-yantings 更新时间：2026-06-22 10:29

```
/**
 * 将第三方服务的自定义数据格式转换为数字孪生服务协议的数据格式，设备、属性上报、事件上报到DTS时调用。
 * @param {object} body - 入参
 * @param {object} body._bv_callback_message - 服务回调参数，回调时存在
 * @param {string} body._bv_callback_message.productId - 产品编号
 * @param {string} body._bv_callback_message.deviceId - 设备编号
 * @param {string} body._bv_callback_message.identifier - 事件的唯一标识符，回调方式为事件上报时存在
 * @param {object} body._bv_callback_message._bv_service_callback - 服务调用参数
 * @param {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * @param {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识
 * @param {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * @param {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * @param {number} flag - 入参，0 设备上报 1 属性上报 2 事件上报
 * @returns {object} 出参
 */
function rawDataToProtocol(body, flag) {
    var jsonMap = {};
    var inputData = body;
    if (flag === 0) {
        /**
         * @returns {string} return.deviceId - 设备编号（保证唯一），必传
         * @returns {string} return.deviceName - 设备名称（最大长度255），必传
         * @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
         * @returns {number} return.deviceType - 设备类型 0:动态资源 1:静态资源，必传
         * @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
         * @returns {string} return.version - 版本号，默认default，非必传
         * @returns {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义的属性进行过滤，已存在定义的数据会保留，
         */
        return body;
    }
    if (flag === 1) {
        /**
         * @returns {string} return.deviceId - 设备编号（保证唯一），必传
         * @returns {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
         */
    }
}
```

```

* @returns {string} return.deviceName - 设备名称（最大长度255），非必传
* @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
* @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
* @returns {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数，非必传
* @returns {string} return.version - 版本号，默认default，非必传
* @returns {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义的属性进行过滤，已存在定义的数据会保留，
*/
if (inputData._bv_callback_message) {
  var bv_callback_message = inputData._bv_callback_message;
  var bv_service_callback = bv_callback_message._bv_service_callback;
  var payload = inputData.payload ? inputData.payload : {};
  if (bv_service_callback.identifier == "captureServerImage" || bv_service_callback.identifier == "captureServerVideo") {
    if (inputData.result != 0) {
      return;
    }
    if (bv_service_callback.identifier == "captureServerImage") {
      var resourceUrl = FileStorage.cache(inputData.file, inputData.suffix, bv_callback_message.deviceId);
      payload.deviceImage = resourceUrl;
    }
    if (bv_service_callback.identifier == "captureServerVideo") {
      var resourceUrl = FileStorage.disk(inputData.file, inputData.suffix);
      payload.deviceVideoUrl = resourceUrl;
    }
  }
  if (bv_service_callback.identifier == "video_attractive_score") {
    payload.score = inputData.score;
  }
  if (bv_service_callback.identifier == "captureImage") {
    //payload.deviceImage = inputData.payload.deviceImage.replace("172.27.240.1","10.132.25.2");
  }
  jsonMap.deviceId = bv_callback_message.deviceId;
  jsonMap.payload = payload;

  return jsonMap;
} else {
  return body;
}
}
if (flag === 2) {
  /**
  * @returns {string} return.deviceId - 设备编号（保证唯一），必传
  * @returns {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
  * @returns {string} return.deviceName - 设备名称（最大长度255），非必传
  * @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
  * @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
  * @returns {string} return.identifier - 事件的唯一标识符，必传
  * @returns {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数，非必传
  * @returns {string} return.version - 版本号，默认default，非必传
  * @returns {Object} return.payload - 扩展数据，事件的输出参数（该数据根据产品Profile中定义的事件输出参数进行过滤，已存在定义的数据会保留，其余数据
  */
  if (inputData._bv_callback_message) {
    jsonMap['deviceId'] = inputData._bv_callback_message.deviceId;
    jsonMap['identifier'] = inputData._bv_callback_message.identifier;
    // 添加mask过滤
    var entityAttribute = CoreApi.getEntityAttribute(inputData._bv_callback_message.productId, inputData._bv_callback_message.deviceId);
    entityAttribute = JSON.parse(entityAttribute);
    if (typeof entityAttribute === 'object' && entityAttribute !== null && entityAttribute.hasOwnProperty('TestImageMask') && entityAttribute.hasOwnProperty('devi
      const mask = JSON.parse(entityAttribute.TestImageMask);
      inputData = maskFilter(inputData, mask)
      inputData.boundingBoxImage = drawBox(inputData, entityAttribute.deviceImage)

    }

    // 存在文件资源时将存储至网关服务下
    if (inputData.heatMapImage) {
      inputData.heatMapImageUrl = FileStorage.disk(inputData.heatMapImage, "jpg");
    }
    if (inputData.boundingBoxImage) {
      inputData.boundingBoxImageUrl = FileStorage.disk(inputData.boundingBoxImage, "jpg");
    }
    jsonMap['payload'] = {
      "analysisTime": inputData.analysisTime,
      "objectNumber": inputData.objectNumber,
      "objectBoundingBox": inputData.objectBoundingBox,
      "boundingBoxImageUrl": inputData.boundingBoxImageUrl,
      "heatMapImageUrl": inputData.heatMapImageUrl
    }
  }
  return jsonMap;
}
}
//在图片上绘制分析结果框
function drawBox(box,image) {
  const labels = [];
  const boxes = [];
  Object.keys(box.objectBoundingBox).forEach(function (key) {

```

```
    const value = box.objectBoundingBox[key];
    boxes.push(JtsGeometry.createCoordinatesXXYY(value));
    labels.push(key);
  });
  return JtsGeometry.drawBox(image, boxes, "Number: " + box.objectNumber, labels);
}

/**
 * 根据掩码 (mask) 过滤 box 对象中的边界框信息。
 * @param {object} inputData - 包含边界框信息的对象，其中 objectBoundingBox 属性存储具体的边界框数据。
 * @param {object} mask - 掩码对象，包含几何信息用于与边界框进行交集判断。
 */
function maskFilter(inputData, mask) {
  if (typeof inputData.objectBoundingBox !== 'object' && inputData.objectBoundingBox === null) {
    console.log("inputData.objectBoundingBox 不是有效的对象")
    return inputData;
  }
  const in_mask = mask.properties.mask_type === "positive";
  var objectBoundingBoxObj = {};
  var objectNumber = 0;
  for (const key in inputData.objectBoundingBox) {
    let value = inputData.objectBoundingBox[key];

    // value保留三位小数 double类型
    for (let i = 0; i < value.length; i++) {
      if (typeof value[i] !== 'number') {
        value[i] = parseFloat(value[i].toString());
      }
      value[i] = parseFloat(value[i].toFixed(3));
    }

    // 使用 JtsGeometry 工具类将边界框坐标转换为多边形所需的坐标数组
    const boxCoord = JtsGeometry.createCoordinatesXXYY(value);
    // 创建边界框对应的多边形
    let boxPolygon = JtsGeometry.createPolygon(boxCoord);
    for (let i = 0; i < mask.geometry.annotations.length; i++) {
      const maskCoord = JtsGeometry.createCoordinates(mask.geometry.annotations[i].coord);
      const maskPolygon = JtsGeometry.createPolygon(maskCoord);
      // 如果 in_mask 为 true 且边界框多边形和掩码多边形有交集
      if (in_mask && JtsGeometry.isIntersects(maskPolygon, boxPolygon)) {
        //console.log("box和mask有交集，交集面积为", JtsGeometry.intersectionArea(maskPolygon, boxPolygon))
        objectBoundingBoxObj[key] = value;
        objectNumber++;
        break;
      }
      // 如果 in_mask 为 false 且边界框多边形和掩码多边形没有交集
      if (!in_mask && !JtsGeometry.isIntersects(maskPolygon, boxPolygon)) {
        //console.log("负掩码，box和mask没有交集即为有效区域")
        objectBoundingBoxObj[key] = value;
        objectNumber++;
        break;
      }
    }
  }
  inputData.objectBoundingBox = objectBoundingBoxObj;
  inputData.objectNumber = objectNumber;
  return inputData;
}
```

作者：北京-产品-yantings 创建时间：2025-05-28 14:08 更新时间：2025-08-19 15:31 The above content shows the entire, complete file contents of the requested file.

AI算法的快速应用落地

AI算法的快速应用落地

本文将为你介绍使用DTS调度AI算法服务的流程，实现AI算法的快速应用落地。首先需要先把AI算法进行服务化，然后在DTS的产品-服务中进行定义，接下来我们就可以在任务管理中快速进行调度和使用了。

包含以下几个流程：

网关添加服务地址、创建产品、定义产品Profile、创建任务、结果验证

示例通过从摄像头定时获取图片进行人数分析的流程演示整个过程。

准备事项

- 1.注册数字孪生服务DTService（以下简称DTS）账户
- 2.准备APIfox，用于模拟设备（或其他可以发送Http请求的软件，如postman、vs等）。实际使用时，设备需支持反向控制。
- 3.成功部署看门狗服务，连接实时摄像头或演示视频数据；
- 4.成功把AI算法进行服务化

示例

AI算法服务化

此步骤为简单可实现的AI算法服务化方法，可根据实际使用需要进行调整或修改。

1. 在服务器或本地机器上安装Python 3.8+环境。然后使用pip安装核心库，包括Ultralytics（用于YOLO模型）、FastAPI（用于构建API）、Uvicorn（作为API服务器）、以及OpenCV或Pillow（用于图片处理）。
2. 从Ultralytics官网下载预训练YOLO模型（如yolov11.pt），或使用自定义训练的模型。通过Ultralytics API加载模型（例如model = YOLO('yolov11n.pt')），确保模型能处理图片输入并输出检测结果（如边界框、类别、置信度）。
3. 定义图片输入格式（支持JPEG/PNG等），并准备输出格式（如JSON，包含检测结果）。
4. 使用FastAPI创建一个简单的Web应用。定义一个POST端点（如/predict），允许用户上传资源文件。
5. 在端点处理函数中，接收上传的资源（如图片），传入YOLO模型进行推理（例如调用model.predict(image)），提取结果（如对象位置和标签），并转换为JSON响应返回给调用者。

DTS 作为调用者且使用异步调用时，会发送[回执参数](#)以及输入参数到AI算法服务端。AI算法服务端需根据回执参数中的结果返回地址和方式将结果发送到对应目标地址。DTS收到分析结果，才能正确解析。结果返回的方式，DTS使用者和AI算法提供者双方可以协商确定。
同步调用时，AI算法可直接返回结果，DTS使用者根据自己需要决定结果返回方式。结果格式不满足BV JSON时，需要编写消息解析脚本进行解析。

1. 使用Uvicorn启动API服务器（命令如uvicorn main:app --reload），在本地浏览器或Postman工具中测试：上传图片，调用API，检查返回结果是否正确（例如JSON中包含检测到的对象信息）。

2. 使用Docker将应用打包成镜像（编写Dockerfile，包括安装依赖和复制代码），便于移植。构建镜像并测试容器运行。

本示例中采用YOLO11模型为基础搭建AI算法服务器，将结果以事件上报的方式返回到DTS中。

部分代码示例

获取回执参数和回调地址

```
# 记录请求内容
data = request.get_json()
...
callback_message = data.get('_bv_callback_message', {})
callback_url = data.get('_bv_callback_url', "")
```

发送结果时携带回调参数

```
# 如果提供了回调信息
if callback_message:
    callback_result['_bv_callback_message'] = callback_message
....
# 如果提供了回调URL，发送POST请求
if callback_url:
    try:
        requests.post(callback_url, json=callback_result, timeout=5)
        logger.info(f"[回调] 发送分析结果到 {callback_url} 成功")
    except Exception as e:
        logger.error(f"[回调] 发送分析结果到 {callback_url} 失败: {e}")
```

网关

添加服务地址

1. 进入实体管理模块，选择网关。
2. 选择“通用网关”，点击编辑进入通用网关的详情页面->选择服务列表 Tab页面。
3. 在服务列表 Tab页面下，选择添加一个服务地址。
4. 命名为“AICenterServer”，选择类型为Http方式，服务地址输入指向AI算法服务化后的地址。

这个服务地址需指向接收控制服务请求的目标地址。示例选择使用Http方式，还可以根据需要选择WebSocket、Redis等方式。



创建产品-AI产品

1. 进入实体管理模块，选择产品。
2. 选择创建一个新的产品。
3. 命名为“AI产品”，设定接入网关协议选择为HTTP协议，数据格式为透传/自定义。

4. 设置完成后点击立即创建。创建成功的产品，会自动创建品类中的产品Profile内容。

加入网关

1. 进入实体管理模块，选择网关。
2. 选择“通用网关”，点击编辑进入网关的详情页面->选择网关信息 Tab页面。
3. 点击编辑，进入网关信息编辑页面。
4. 修改关联产品，勾选中“AI产品”产品。将新创建的产品加入到**通用网关**下。

通用网关为系统内置的网关，可支持加入HTTP协议接入的实体。

定义产品Profile（事件）

1. 进入实体管理模块，选择产品。
2. 选择“AI产品”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为**事件**。
4. 输入功能名称为“分析结果”，设置标识符为analysis_count，事件类型选择为告警。
5. 根据AI算法服务化的算法返回的分析结果设置输出参数。此处示例定义输出参数 `objectNumber` `objectBoundingBox` `boundingBoxImage` `boundingBoxImageUrl` `time` `heatMapImageUrl` `heatMapImage`。
6. 根据需要选择是否需要持久化存储，此处示例勾选需要。
7. 点击确定保存事件。

需根据具体使用的AI算法服务化的算法进行相关定义。

* 功能类型

事件

* 功能名称

分析结果

* 标识符

analysis_count

* 事件类型

信息

告警

* 消息质量

默认

一级

二级

输出参数

参数名称: objectNumber

编辑 | 删除

参数名称: objectBoundingBox

编辑 | 删除

参数名称: boundingBoxImage

编辑 | 删除

参数名称: boundingBoxImageUrl

编辑 | 删除

参数名称: time

编辑 | 删除

参数名称: heatMapImageUrl

编辑 | 删除

参数名称: heatMapImage

编辑 | 删除

+ 增加参数

操作

持久化存储

确定

取消

内容窗口 - ... (服务)

本文档由北京大视景科技有限公司生成- 175 -

定义产品Profile（服务）

1. 进入实体管理模块，选择产品。
2. 选择“AI产品”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择添加功能，选择功能类型为**服务**。
4. 输入功能名称为“图片分析”，设置标识符为picture_analysis_count。
5. 服务地址选择已定义的服务地址“AICenterServer”。
6. 根据**AI算法服务化的算法所需参数**，服务设置输入参数 `analysisType` `resultType` `analysisResource`，输出参数 `analysisTime`。
7. 选择结果返回方式为“事件上报”，反控成功后，通过目标事件分析结果的事件返回。

设备反控执行后，根据使用场景选择结果返回方式，是否需要将服务执行结果反馈到DTS中或其他第三方服务中。在产品Profile所有编辑完成后，需点击**发布**。发布后的产品Profile才能被DTS识别。结果返回方式选择“属性上报”时，需定义相应的属性。可参考定义产品Profile（事件）

编辑消息解析脚本

1. 进入实体管理模块，选择产品。
2. 选择“AI产品”产品，点击编辑进入产品的详情页面->选择消息解析 Tab页面。
3. 选择实体映射（上行）Tab页面，编辑上行解析脚本；

```
/**
 * 将第三方服务的自定义数据格式转换为数字孪生服务协议的数据格式，设备、属性上报、事件上报到DTS时调用。
 * [@param](https://github.com/param "@param") {object} body - 入参
 * [@param](https://github.com/param "@param") {object} body._bv_callback_message - 服务回调参数，回调时存在
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.productId - 产品编号
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.deviceId - 设备编号
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.identifier - 事件的唯一标识符，回调方式为事件上报时存在
 * [@param](https://github.com/param "@param") {object} body._bv_callback_message._bv_service_callback - 服务调用参数
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * [@param](https://github.com/param "@param") {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * [@param](https://github.com/param "@param") {number} flag - 入参，0 设备上报 1 属性上报 2 事件上报
 * [@returns](https://github.com/returns "@returns") {object} 出参
 */
function rawDataToProtocol(body, flag) {
  var jsonMap = {};
  var inputData = body;
  if (flag === 0) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceType - 设备类型 0:动态资源 1:静态资源，必传
     * [@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.version - 版本号，默认default，非必传
     * [@returns](https://github.com/returns "@returns") {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义）
     */
    return body;
  }
  if (flag === 1) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分
     * [@returns](https://github.com/returns "@returns") {string} return.version - 版本号，默认default，非必传
     * [@returns](https://github.com/returns "@returns") {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义）
     */
    return body;
  }
  if (flag === 2) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.identifier - 事件的唯一标识符，必传
     * [@returns](https://github.com/returns "@returns") {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分
     * [@returns](https://github.com/returns "@returns") {string} return.version - 版本号，默认default，非必传
     */
  }
}
```

```

*[@returns]([https://github.com/returns">@returns") {Object} return.payload - 扩展数据, 事件的输出参数 (该数据根据产品Profile中定义的事件输出参数进行)
*/
if (inputData._bv_callback_message) {
    jsonMap['deviceId'] = inputData._bv_callback_message.deviceId;
    jsonMap['identifier'] = inputData._bv_callback_message.identifier;
    // 存在文件资源时将存储至网关服务下
    if(inputData.heatMapImage){
        inputData.heatMapImageUrl = FileStorage.disk(inputData.heatMapImage, "jpg");
    }
    if(inputData.boundingBoxImage){
        inputData.boundingBoxImageUrl = FileStorage.disk(inputData.boundingBoxImage, "jpg");
    }
    jsonMap['payload'] = {
        "analysisTime": inputData.analysisTime,
        "objectNumber": inputData.objectNumber,
        "objectBoundingBox": inputData.objectBoundingBox,
        "boundingBoxImageUrl": inputData.boundingBoxImageUrl,
        "heatMapImageUrl": inputData.heatMapImageUrl
    }
}
return jsonMap;
}
}

```

创建虚拟实体

1. 进入实体管理模块，选择实体。
2. 选择“添加实体”，为AI产品创建一个虚拟实体。
3. 在添加实体页面，选择产品为“AI产品”，DeviceID选择自动生成一个DeviceID，输入实体名称为“AI虚拟实体”。
4. 输入完成点击确定。DTS创建一个新的实体。

添加实体

特别说明： DeviceID可以为空，当为空时，数字孪生模块会颁发全局唯一标识符作为DeviceID，网关可以不选，后期可在产品实体列表进行绑定。

★ 产品

AI产品

网关

请选择网关

DeviceID

I1sXdELchIB0Pskt

自动生成

1. 进入实体管理模块，选择产品。
2. 选择创建一个新的产品。
3. 命名为“BV 相机”，从**标准品类** 中选择产品名称为“BV相机”的产品，设定接入网关协议选择为HTTP协议，数据格式为透传/自定义。
4. 设置完成后点击立即创建。创建成功的产品，会自动创建品类中的产品Profile内容。

标准品类中BV相机已经预先定义完成产品Profile，其中包括bv相机的属性、事件以及所能支持的视频操作服务。

1. 进入实体管理模块，选择产品。
2. 选择创建一个新的产品。
3. 命名为“BV 相机”，选择自定义品类，设定接入网关协议选择为HTTP协议，数据格式为透传/自定义。
4. 设置完成后点击立即创建。

1. 进入实体管理模块，选择产品。
2. 选择“BV 相机”产品，点击编辑进入产品的详情页面->选择产品Profile Tab页面。
3. 点击编辑草稿，选择导入产品Profile功能；
4. 下载产品Profile文件保存到本地，并将后缀修改为“json”。
5. 导入文件选择预先存在本地的产品Profile文件，选择导入。
6. 提示导入成功，可查看到导入的属性、事件、服务的定义。

产品Profile内容可能会有因为版本的不同存在定义上的差异。

← BV相机					
属性	服务请求编号 可选	requestId	text(字符串)	数据长度:255	详情
属性	客户端Token 可选	clientToken	text(字符串)	数据长度:255	详情
属性	扩展数据 自定义	extendData	text(字符串)	数据长度:128	详情
属性	媒体流ID/token 可选	mediaToken	text(字符串)	数据长度:255	详情
属性	媒体源编号 可选	mediaSourceGuid	text(字符串)	数据长度:128	详情
属性	设备编号 自定义	deviceGuid	text(字符串)	数据长度:128	详情
属性	设备类型 自定义	deviceType	enum(枚举型)	0-主摄; 1-相机; 2-球机; 3-鱼眼;	详情
属性	设备名称 自定义	deviceName	text(字符串)	数据长度:128	详情
服务	车辆识别计数服务 自定义	vehicle_detect_count	-		详情
服务	打开视频流 自定义	openMedia	-		详情
服务	关闭视频流 自定义	closeMedia	-		详情
服务	获取ptz位置 自定义	getPtzPos	-		详情
服务	获取视频片段 自定义	getVideoFragment	-		详情
服务	获取视频设备列表 自定义	getMediaDeviceList	-		详情

编辑消息解析脚本

1. 进入实体管理模块，选择产品。
2. 选择“BV 相机”产品，点击编辑进入产品的详情页面->选择消息解析 Tab页面。
3. 选择实体映射（上行）Tab页面，编辑上行解析脚本；

```

/**
 * 将第三方服务的自定义数据格式转换为数字孪生服务协议的数据格式，设备、属性上报、事件上报到DTS时调用。
 * [@param](https://github.com/param "@param") {object} body - 入参
 * [@param](https://github.com/param "@param") {object} body._bv_callback_message - 服务回调参数，回调时存在
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.productId - 产品编号
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.deviceId - 设备编号
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message.identifier - 事件的唯一标识符，回调方式为事件上报时存在
 * [@param](https://github.com/param "@param") {object} body._bv_callback_message._bv_service_callback - 服务调用参数
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识
 * [@param](https://github.com/param "@param") {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * [@param](https://github.com/param "@param") {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * [@param](https://github.com/param "@param") {number} flag - 入参，0 设备上报 1 属性上报 2 事件上报
 * [@returns](https://github.com/returns "@returns") {object} 出参
 */
function rawDataToProtocol(body, flag) {
  var jsonMap = {};
  var inputData = body;
  if (flag === 0) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceType - 设备类型 0:动态资源 1:静态资源，必传
     * [@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.version - 版本号，默认default，非必传
     * [@returns](https://github.com/returns "@returns") {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义）
     */
    return body;
  }
  if (flag === 1) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * [@returns](https://github.com/returns "@returns") {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分
     * [@returns](https://github.com/returns "@returns") {string} return.version - 版本号，默认default，非必传
     * [@returns](https://github.com/returns "@returns") {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义）
     */
    if (inputData._bv_callback_message) {
      var bv_callback_message = inputData._bv_callback_message;
      var bv_service_callback = bv_callback_message._bv_service_callback;
      var payload = inputData.payload ? inputData.payload : {};
      if (bv_service_callback.identifier == "captureServerImage" || bv_service_callback.identifier == "captureServerVideo") {
        if (inputData.result != 0) {
          return;
        }
        if (bv_service_callback.identifier == "captureServerImage") {
          var resourceUrl = FileStorage.cache(inputData.file, inputData.suffix, bv_callback_message.deviceId);
          payload.deviceImage = resourceUrl;
        }
        if (bv_service_callback.identifier == "captureServerVideo") {
          var resourceUrl = FileStorage.disk(inputData.file, inputData.suffix);
          payload.deviceVideoUrl = resourceUrl;
        }
      }
      if (bv_service_callback.identifier == "video_attractive_score") {
        payload.score = inputData.score;
      }
      if (bv_service_callback.identifier == "captureImage") {
        //payload.deviceImage = inputData.payload.deviceImage.replace("172.27.240.1","10.132.25.2");
      }
      jsonMap.deviceId = bv_callback_message.deviceId;
      jsonMap.payload = payload;
      return jsonMap;
    } else {
      return body;
    }
  }
  if (flag === 2) {
    /**
     * [@returns](https://github.com/returns "@returns") {string} return.deviceId - 设备编号（保证唯一），必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
     * [@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称（最大长度255），非必传

```

```

    *[@returns](https://github.com/returns "@returns") {string} return.deviceName - 设备名称 (默认长度255) , 非必填
    *[@returns](https://github.com/returns "@returns") {number} return.deviceStatus - 设备状态 0:禁用 1:启用 (默认) , 非必填
    *[@returns](https://github.com/returns "@returns") {string} return.location - 设备位置经纬度 (默认0 0 0 = latitude longitude elevation) , 非必填
    *[@returns](https://github.com/returns "@returns") {string} return.identifier - 事件的唯一标识符, 必填
    *[@returns](https://github.com/returns "@returns") {number} return.timestamp - 上行数据传输时间。时间戳格式为数值, 值为自GMT 1970年1月1日0时0分
    *[@returns](https://github.com/returns "@returns") {string} return.version - 版本号, 默认default, 非必填
    *[@returns](https://github.com/returns "@returns") {Object} return.payload - 扩展数据, 事件的输出参数 (该数据根据产品Profile中定义的事件输出参数进行)
    */
    if (inputData._bv_callback_message) {
        jsonMap['deviceId'] = inputData._bv_callback_message.deviceId;
        jsonMap['identifier'] = inputData._bv_callback_message.identifier;
        // 存在文件资源时将存储至网关服务下
        if(inputData.heatMapImage){
            inputData.heatMapImageUrl = FileStorage.disk(inputData.heatMapImage, "jpg");
        }
        if(inputData.boundingBoxImage){
            inputData.boundingBoxImageUrl = FileStorage.disk(inputData.boundingBoxImage, "jpg");
        }
        jsonMap['payload'] = {
            "analysisTime": inputData.analysisTime,
            "objectNumber": inputData.objectNumber,
            "objectBoundingBox": inputData.objectBoundingBox,
            "boundingBoxImageUrl": inputData.boundingBoxImageUrl,
            "heatMapImageUrl": inputData.heatMapImageUrl
        }
    }
    return jsonMap;
}
}
```

加入网关

1. 进入实体管理模块，选择网关。
2. 选择“BVMedia网关”，点击编辑进入网关的详情页面->选择网关信息 Tab页面。
3. 点击编辑，进入网关信息编辑页面。
4. 修改关联产品，勾选中BV 相机产品。将新创建的产品加入到**BVMedia网关** 下。

BVMedia网关为DTS内置视频网关，用于接入来自看门狗服务下的实体，无需开发和创建。BVMedia网关会定时更新看门狗服务中的视频设备到网关关联的产品下。更新之后，BV 相机产品就会有对应的实体。此过程需耐心等待。如想快速加入实体，可使用**从网关发现实体** 功能。

从网关发现实体

接口“获取相机设备集合”为系统预设置的接口，用于从看门狗服务下请求设备。2025年4月10日之后的DTS版本支持。

1. 进入实体管理模块，选择实体。
2. 选择功能批量添加，进入批量添加实体页面；
3. 选择产品为“BV相机”，使用“从网关发现实体”的添加方式，选择请求接口“获取相机设备集合”。
4. 设置完成，选择下一步。
5. 系统会请求接口获取接口下的设备实体，若“提示发现X个实体”表示接口请求成功，点击确定即可添加对应的实体到产品下。若“提示发现0个实体”表示接口未能获取到实体，请检查接口。



自动生成

批量上传

从网关发现实体

网关

BVMedia

请求接口

获取相机设备集合

下一步

创建任务

1. 进入任务管理模块，选择任务。
2. 在任务管理页面，创建一个任务名为“人数统计”的任务。

任务逻辑

每半小时 触发一次分析
执行任务 看门狗下的视频设备截取一张图用来做AI分析
AI分析获取到截图执行分析

1. 创建完成后，选择“人数统计”任务点击编辑，进入编辑页面。
2. 根据任务逻辑编辑对应的规则。设定触发条件为每小时，添加执行任务类型为“服务调用”，选择BV 相机产品下的某一设备或全部设备作为执行对象，选择服务为“异步抓图服务”，并点击参数配置，设置服务调用需要的输入参数。设置“设备出厂编号”的参数值为当前实体的出厂编号属性值。

异步抓图服务调用参数配置

参数配置

配置服务调用时对应的输入参数的参数值,参数值可从多种来源中选择。请根据服务参数数据类型匹配正确的参数,系统将自动获取该参数值。

服务调用

异步抓图服务

输入参数

选择参数值进行配置,建立映射

text(字符串)

设备出厂编号(cameraFactoryGuid)

BV相机公司环境...

text(字符串)媒体源编号(mediaSourceGuid)

输入参数

text(字符串)设备编号(mediaDeviceGuid)

输入参数

text(字符串)utc 开始时间 (毫秒值) (beginTimestamp)

输入参数

text(字符串)expiredTimestamp(expiredTimestamp)

输入参数

int32(整数值)captureRetryLimit(captureRetryLimit)

输入参数

int32(整数值)callBackRetryLimit(callBackRetryLimit)

输入参数

text(字符串)format(format)

输入参数

text(字符串)抓图服务请求地址(requestUrl)

输入参数

取消确定

1. 可在下一次任务调用前设置一定时间的延迟，添加一个“延迟”任务。
2. 添加执行任务类型为“服务调用”，选择“AI产品”产品下的某一设备或全部设备作为执行对象，选择服务为“图片分析”并点击参数配置，设置服务调用需要的输入参数。

图片分析服务调用参数配置

支持的分析类型以及编码 {0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking meter'}

返回结果字段及其编码 {1: 'objectNumber', 2: 'objectBoundingBox', 4: 'heatMapImage', 8: 'boundingBoxImage', 16: 'boundingBoxImageUrl', 32: 'heatMapImageUrl'}

示例中分析图片来源为执行截图任务的实体的当前截图地址；分析类型为person，返回结果字段为objectNumber、objectBoundingBox、boundingBoxImage

参数配置

配置服务调用时对应的输入参数的参数值,参数值可从多种来源中选择。请根据服务参数数据类型匹配正确的参数,系统将自动获取该参数值。

服务调用 图片分析_http

服务调用

服务调用

输入参数 选择参数值进行配置，建立映射

array(数组)

analysisType(analysisType)

[0]

+

int32(整数型)

resultType(resultType)

11

+

text(字符串)

analysisResource(analysisResource)

BV相机公司环境...

+

+

取消

确定

1. 配置完成，保存并启动任务。

如果满足一下任一触发条件 ? + 添加触发条件

定时触发

每天

14:30:05

选择器

且满足以下全部生效条件 + 添加生效条件

就执行以下任务 + 添加执行任务

1. 设备控制-服务调用



结果验证

前往任务-执行日志，查看任务的执行情况。

任务管理 / 任务 / 执行日志

← 执行日志

全部状态

定时车辆识别计数-识别数据-自动

自定义

2025-02-08 14:54:16 - 2025-03-10 14:54:16

模块名称	执行类型	执行任务	执行时间	执行状态	操作
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 11:30:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 11:00:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 10:30:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 10:00:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 09:30:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-03 09:00:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-02 19:30:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-02 19:00:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-02 18:30:00	成功	详情
定时车辆识别计数-识别数据-自动	定时执行	设备控制 - 服务调用、设备控制 - 服务调用、设备控制 - 服务调用	2025-03-02 18:00:00	成功	详情

数据监控

经过[订阅管理](#)将向DTS订阅设备的属性变化和事件发生的数据，订阅端收到数据之后进行可视化处理便能对设备进行实时数据监控。DTS数据孪生架构提供快速接入查看DTS设备数据的平台功能[低代码设计器](#)（无需订阅），详细可视化功能可查看[《BigView低代码设计器教程》](#)

附件

- [BV相机.txt](#)

附件

- [BV相机.txt](#)

附录：BV相机产品Profile介绍

BV相机产品Profile介绍

本文介绍标准品类中BV相机的产品Profile。BV相机来源于看门狗服务系统，是适配于看门狗的品类。该产品主要涉及视频监控、媒体流处理及分析服务。

基本信息

- 产品品类：BV相机

属性

属性定义了产品或设备的特性。每个属性包括名称、标识符、数据类型、读写权限、是否必须以及添加时间等信息。读写权限：0（只读），1（读写）。是否必须：0（可选），2（必须）。

属性名称	标识符	数据类型	读写权限	是否必须
设备编号	deviceGuid	text(字符串)	1	2
测试图片掩膜	TestImageMask	ImageMask(图片掩膜)	0	2
吸引力得分	score	int32(整数型)	0	2
媒体源编号	mediaSourceGuid	text(字符串)	0	0
出厂编号	factoryGuid	text(字符串)	0	0
dash流媒体访问地址	dashMediaUrl	text(字符串)	0	0
hls流媒体访问地址	hlsMediaUrl	text(字符串)	0	0
rtmp流媒体访问地址	rtmpMediaUrl	text(字符串)	0	0
rtsp流媒体访问地址	rtspMediaUrl	text(字符串)	0	0
媒体流访问token	mediaToken	text(字符串)	0	0
客户端token	clientToken	text(字符串)	0	0
当前设备视频片段	deviceVideoUrl	text(字符串)	0	0
设备名称	deviceName	text(字符串)	1	2

属性名称	标识符	数据类型	读写权限	是否
设备类型	deviceType	enum(枚举型)	1	2
扩展数据	extendData	text(字符串)	1	2
当前设备截图地址	deviceImage	text(字符串)	1	0
服务请求编号	requestId	text(字符串)	1	0

事件

事件定义了通知或回调机制。事件类型：0（信息），1（告警）。QoS：0（最多一次）。持久化存储：0（无），1（有）。是否必须：2（必须）。

设备信息更新

- 标识符：deviceInfo
- 类型：0
- 是否必须：2
- 持久化存储：1
- QoS：0
- 输出数据：
 - deviceImage: text(字符串), 长度=255 - 设备截图
 - requestId: text(字符串), 长度=255 - 请求ID

车辆识别计数

- 标识符：vehicle_count
- 类型：1
- 是否必须：2
- 持久化存储：0
- QoS：0
- 描述：用于服务调用后回调
- 输出数据：
 - objectNumber: int32(整数型), min=1, max=111 - 目标数量
 - objectBoundingBox: text(字符串), 长度=255 - 目标边界框
 - boundingBoxImage: text(字符串), 长度=512 - 边界框图片
 - boundingBoxImageUrl: text(字符串), 长度=128 - 边界框图片URL
 - analysisTime: text(字符串), 长度=128 - 分析时间

行人识别计数

- 标识符 : crowd_count
- 类型 : 1
- 是否必须 : 2
- 持久化存储 : 0
- QoS : 0
- 描述 : 用于服务调用后回调
- 输出数据 :
 - objectNumber: int32(整数型), min=1, max=111 - 目标数量
 - objectBoundingBox: text(字符串), 长度=255 - 目标边界框
 - boundingBoxImage: text(字符串), 长度=512 - 边界框图片
 - boundingBoxImageUrl: text(字符串), 长度=128 - 边界框图片URL
 - analysisTime: text(字符串), 长度=128 - 分析时间
 - heatMapImageUrl: text(字符串), 长度=128 - 热力图URL
 - heatMapImage: text(字符串), 长度=512 - 热力图

服务

服务定义了可调用的操作，包括输入/输出数据、结果返回方式等配置。结果返回方式：0（无），2（HTTP），3（Redis），4（属性），5（事件）。调用方式：0（同步），1（异步）。部分服务会关联事件用于回调。

异步视频服务_代理

- 标识符 : captureServerVideoProxy
- 描述 : (空)
- 结果返回方式 : 4
- 调用方式 : 1
- 接口 URL : /api/accessGateway/captureServerVideo
- 输入数据 :
 - cameraFactoryGuid: text(字符串), 长度=128 - 设备出厂编号
 - mediaSourceGuid: text(字符串), 长度=128 - 媒体源编号
 - mediaDeviceGuid: text(字符串), 长度=128 - 设备编号
 - beginTimestamp: text(字符串), 长度=128 - UTC 开始时间（毫秒值），-1=取实时码流（受任务队列影响），历史抓取只支持当前时间5分钟之前的历史数据
 - expiredTimestamp: text(字符串), 长度=128 - 过期时间，-1=不超时，可以使用相对时间，+10s表示10s后，-10s表示10秒前，+500ms表示500毫秒后，-500ms表示500毫秒前
 - captureRetryLimit: int32(整数型) - 捕获重试限制，不设置则按照服务器默认配置（默认为3）
 - callBackRetryLimit: int32(整数型) - 回调重试限制，不设置则按照服务器默认配置（默认为3）
 - format: text(字符串), 长度=128 - 格式，不设置则按照服务器默认配置（默认mp4）
 - requestUrl: text(字符串), 长度=255 - 抓图服务请求地址（仅支持 Thrift HTTP 协议）
- 输出数据 :
 - file: text(字符串), 长度=128 - 文件(base64)
 - suffix: text(字符串), 长度=128 - 后缀，jpg、png、MP4、avi等

- result: int32(整数型) - 结果， 0:成功 1:失败

异步抓图服务_代理

- 标识符 : captureServerImageProxy
- 描述 : (空)
- 结果返回方式 : 4
- 调用方式 : 1
- 接口 URL : /api/accessGateway/captureServerImage
- 输入数据 : (与异步视频服务_代理类似, 缺少 duration)
- 输出数据 : (同上)

异步视频服务

- 标识符 : captureServerVideo
- 描述 : (空)
- 结果返回方式 : 4
- 调用方式 : 1
- 接口 URL : (空)
- 输入数据 : (与代理服务类似, 新增 duration: int32(整数型) - 持续时间)
- 输出数据 : (同上)

异步抓图服务

- 标识符 : captureServerImage
- 描述 : (空)
- 结果返回方式 : 4
- 调用方式 : 1
- 接口 URL : (空)
- 输入数据 : (与视频服务类似, 缺少 duration)
- 输出数据 : (同上)

请求截图

- 标识符 : captureImage
- 描述 : 测试属性回调
- 结果返回方式 : 4
- 调用方式 : 0
- 输入数据 :
 - deviceGuid: text(字符串), 长度=32 - 视频设备GUID
 - streamHint: int32(整数型) - 码流类型
 - replayTime: struct(结构体) - 回放起止时间 (开始时间、结束时间, 单位: ms)
 - playType: int32(整数型) - 播放类型
- 输出数据 :

- deviceImage: text(字符串), 长度=2048 - 图片链接

获取视频片段

- 标识符 : getVideoFragment
- 结果返回方式 : 4
- 调用方式 : 0
- 输入数据 :
 - rtspMediaUrl: text(字符串), 长度=255 - RTSP流媒体访问地址
 - duration: int32(整数型) - 持续时间
- 输出数据 :
 - deviceVideoUrl: text(字符串), 长度=255 - 当前设备视频片段链接

关闭视频流

- 标识符 : closeMedia
- 描述 : (空)
- 结果返回方式 : 4
- 调用方式 : 0
- 输入数据 :
 - mediaToken: text(字符串), 长度=255 - 媒体流访问token
 - clientToken: text(字符串), 长度=255 - 客户端token
- 输出数据 :
 - clientToken: text(字符串), 长度=255 - 客户端token
 - mediaToken: text(字符串), 长度=255 - 媒体流访问token
 - rtspMediaUrl: text(字符串), 长度=255 - RTSP流媒体访问地址
 - rtmpMediaUrl: text(字符串), 长度=255 - RTMP流媒体访问地址
 - hlsMediaUrl: text(字符串), 长度=255 - HLS流媒体访问地址
 - dashMediaUrl: text(字符串), 长度=255 - DASH流媒体访问地址

打开视频流

- 标识符 : openMedia
- 结果返回方式 : 4
- 调用方式 : 0
- 输入数据 :
 - deviceId: text(字符串), 长度=255 - 设备ID
- 输出数据 : (与关闭视频流输出类似)

获取视频设备列表

- 标识符 : getMediaDeviceList

- 输入数据：(无)
- 输出数据：
 - deviceGuid: text(字符串), 长度=255 - 视频设备GUID
 - deviceName: text(字符串), 长度=255 - 视频设备名称
 - deviceType: enum(枚举型) - 视频设备类型 (0=未知, 1=枪机, 2=球机, 3=鱼眼, 4=其他)
 - extendData: text(字符串), 长度=2048 - 视频源返回的视频设备详细信息
 - mediaSourceGuid: text(字符串), 长度=255 - 视频设备所属视频源GUID
 - mediaDeviceList: text(字符串), 长度=1024 - 媒体设备列表

获取 PTZ 位置

- 标识符：getPtzPos
- 描述：获取指定球机的PTZ位置信息
- 回调方法：0
- 输入数据：
 - deviceGuid: text(字符串), 长度=255 - 视频设备GUID
- 输出数据：
 - PtzPos: struct(结构体) - PTZ 位置 (ptzPosZ, ptzPosT, ptzPosP: int32, min=1, max=111)

请求云台控制

- 标识符：controlPtz
- 结果返回方式：0
- 输入数据：
 - deviceGuid: text(字符串), 长度=255 - 视频设备GUID
 - ptzCtrl: enum(枚举型) - PTZ 控制命令 (0=未知, 3=启动等)
 - ptzParam: enum(枚举型) - PTZ 参数 (方向、缩放、焦点等)
 - ptzSpeed: int32(整数型), min=1, max=10 - PTZ 控制速度
 - ptzPos: struct(结构体) - PTZ 位置 (ptzPosP, ptzPosZ, ptzPosT: int32, min=0, max=99999999)
 - ptzPresetPos: struct(结构体) - PTZ 预置位 (ptzPresetName: text 长度=255, ptzPresetID: int32 min=0 max=99999999)
- 输出数据：
 - status: bool(布尔型) - 状态

吸引力排序

- 标识符：attractionSort
- 结果返回方式：3
- 回调 URL：211.157.115.125?port=60379&database=&username=&password=NaVe!0-bvc#9@&channel=/WmdLBvoeh5H1IOpe/*test
- 调用方式：1
- 接口 URL：/video_analysis/center_control
- 输入数据：
 - analysisMission: array(数组) of struct(结构体), 大小=10 - 分析任务

- analysisType: array(数组) of text(字符串) - 分析类型
 - callbackURL: text(字符串), 长度=255 - 回调URL
 - messageId: text(字符串), 长度=255 - 消息ID
- 输出数据: (空)

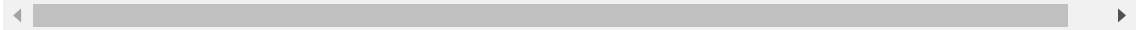
视频吸引力得分

- 标识符: video_attractive_score
- 结果返回方式: 4
- 调用方式: 1
- 接口 URL: /video_attractive_score
- 输入数据:
 - resourceUrl: text(字符串), 长度=128 - 资源URL
- 输出数据:
 - score: int32(整数型) - 吸引力得分

图片分析-ws

- 标识符: picture_analysis_count_ws
 - 结果返回方式: 5
 - 调用方式: 1
 - 接口 URL: /{productId}/{identifier}
 - 目标事件: crowd_count
 - 输入数据:
 - analysisType: array(数组) of int32(整数型), 大小=20 - 分析类型 (如: 0=person, 1=bicycle 等)
 - resultType: int32(整数型) - 结果类型 (位掩码, 支持边界框、热力图等)
 - analysisResource: text(字符串), 长度=1024 - 网络图片或 base64 编码
 - 输出数据: (空)
- 支持的分析类型及其编码

```
{0: 'person', 1: 'bicycle', 2: 'car', 3: 'motorcycle', 4: 'airplane', 5: 'bus', 6: 'train', 7: 'truck', 8: 'boat', 9: 'traffic light', 10: 'fire hydrant', 11: 'stop sign', 12: 'parking me
```



- 支持的结果类型及其编码

```
{1: 'objectNumber', 2: 'objectBoundingBox', 4: 'heatMapImage', 8: 'boundingBoxImage', 16: 'boundingBoxImageUrl', 32: 'heatMapImageUrl'}
```

图片分析-http

- 标识符: picture_analysis_count
- 结果返回方式: 5

- 调用方式：1
- 接口 URL：/picture_analysis_count
- 目标事件：crowd_count
- 输入数据：
 - analysisType: array(数组) of int32(整数型), 大小=20 - 分析类型（如：0=person, 1=bicycle 等）
 - resultType: int32(整数型) - 结果类型（位掩码，支持边界框、热力图等）
 - analysisResource: text(字符串), 长度=1024 - 网络图片或 base64 编码
- 输出数据：(空)
- 支持的分析类型及其编码：（与图片分析-ws一致）
- 支持的结果类型及其编码：（与图片分析-ws一致）

行人识别计数服务

- 标识符：crowd_detect_count
- 描述：请求类型：0x01识别数量 0x02识别框 0x04热力图 0x08边界图 0x10边界图Url 0x20热力图Url
- 结果返回方式：5
- 回调 URL：192.168.100.50:6380&test123
- 调用方式：1
- 接口 URL：
- 目标事件：crowd_count
- 输入数据：
 - requestUrl: text(字符串), 长度=128 - 请求地址
 - requestResultType: text(字符串), 长度=128 - 请求结果类型
 - originalImageUrl: text(字符串), 长度=255 - 原图地址URL
 - deviceName: text(字符串), 长度=128 - 设备名称
- 输出数据：
 - objectNumber: int32(整数型), min=1, max=1111 - 目标数量
 - objectBoundingBox: text(字符串), 长度=255 - 目标边界框
 - boundingBoxImage: text(字符串), 长度=512 - 边界框图片
 - boundingBoxImageUrl: text(字符串), 长度=255 - 边界框图片URL
 - time: text(字符串), 长度=128 - 时间
 - heatMapImageUrl: text(字符串), 长度=128 - 热力图URL
 - heatMapImage: text(字符串), 长度=128 - 热力图

车辆识别计数服务

- 标识符：vehicle_detect_count
- 描述：请求类型：0x01识别数量 0x02识别框 0x04热力图(无) 0x08边界图 0x10边界图Url
- 结果返回方式：5
- 目标事件：vehicle_count
- 输入数据：
 - requestUrl: text(字符串), 长度=128 - 请求地址
 - requestResultType: text(字符串), 长度=128 - 请求结果类型
 - originalImageUrl: text(字符串), 长度=255 - 原图地址URL

- originalImage: text(字符串), 长度=128 - 原图地址URL
- 输出数据：
 - objectNumber: int32(整数型), min=1, max=1111 - 目标数量
 - objectBoundingBox: text(字符串), 长度=255 - 目标边界框
 - boundingBoxImage: text(字符串), 长度=512 - 边界框图片
 - boundingBoxImageUrl: text(字符串), 长度=128 - 边界框图片URL
 - time: text(字符串), 长度=127 - 时间

消息解析脚本

实体映射（上行）

```
/**
 * 将第三方服务的自定义数据格式转换为数字孪生服务协议的数据格式，设备、属性上报、事件上报到DTS时调用。
 * @param {object} body - 入参
 * @param {object} body._bv_callback_message - 服务回调参数，回调时存在
 * @param {string} body._bv_callback_message.productId - 产品编号
 * @param {string} body._bv_callback_message.deviceId - 设备编号
 * @param {string} body._bv_callback_message.identifier - 事件的唯一标识符，回调方式为事件上报时存在
 * @param {object} body._bv_callback_message._bv_service_callback - 服务调用参数
 * @param {string} body._bv_callback_message._bv_service_callback.requestId - 用于标记请求
 * @param {string} body._bv_callback_message._bv_service_callback.identifier - 服务唯一标识
 * @param {string} body._bv_callback_message._bv_service_callback.version - 消息格式版本
 * @param {number} body._bv_callback_message._bv_service_callback.timestamp - 服务调用时间
 * @param {number} flag - 入参，0 设备上报 1 属性上报 2 事件上报
 * @returns {object} 出参
 */
function rawDataToProtocol(body, flag) {
    var jsonMap = {};
    var inputData = body;
    if (flag === 0) {
        /**
         * @returns {string} return.deviceId - 设备编号（保证唯一），必传
         * @returns {string} return.deviceName - 设备名称（最大长度255），必传
         * @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
         * @returns {number} return.deviceType - 设备类型 0:动态资源 1:静态资源，必传
         * @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
         * @returns {string} return.version - 版本号，默认default，非必传
         * @returns {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义的属性进行过滤，已存在定义的数据会保留，
         */
        return body;
    }
    if (flag === 1) {
        /**
         * @returns {string} return.deviceId - 设备编号（保证唯一），必传
         * @returns {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
         * @returns {string} return.deviceName - 设备名称（最大长度255），非必传
         * @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
         * @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
         * @returns {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数，非必传
         * @returns {string} return.version - 版本号，默认default，非必传
         * @returns {Object} return.payload - 扩展数据，产品属性，可用于更新设备属性值（该数据根据产品Profile中定义的属性进行过滤，已存在定义的数据会保留，
         */
        if (inputData._bv_callback_message) {
            var bv_callback_message = inputData._bv_callback_message;
            var bv_service_callback = bv_callback_message._bv_service_callback;
            var payload = inputData.payload ? inputData.payload : {};
            if (bv_service_callback.identifier == "captureServerImage" || bv_service_callback.identifier == "captureServerVideo") {
                if (inputData.result != 0) {
                    return;
                }
            }
            if (bv_service_callback.identifier == "captureServerImage") {
                var resourceUrl = FileStorage.cache(inputData.file, inputData.suffix, bv_callback_message.deviceId);
                payload.deviceImage = resourceUrl;
            }
            if (bv_service_callback.identifier == "captureServerVideo") {
                var resourceUrl = FileStorage.disk(inputData.file, inputData.suffix);
                payload.deviceVideoUrl = resourceUrl;
            }
        }
        if (bv_service_callback.identifier == "video_attractive_score") {

```

```

        payload.score = inputData.score;
    }
    if (bv_service_callback.identifier == "captureImage") {
        //payload.deviceImage = inputData.payload.deviceImage.replace("172.27.240.1","10.132.25.2");
    }
    jsonMap.deviceId = bv_callback_message.deviceId;
    jsonMap.payload = payload;
    return jsonMap;
} else {
    return body;
}
}

if (flag === 2) {
    /**
     * @returns {string} return.deviceId - 设备编号（保证唯一），必传
     * @returns {string} return.deviceSecret - 设备签名（不传时将系统匹配），非必传
     * @returns {string} return.deviceName - 设备名称（最大长度255），非必传
     * @returns {number} return.deviceStatus - 设备状态 0:禁用 1:启用（默认），非必传
     * @returns {string} return.location - 设备位置经纬度（默认0 0 0 = latitude longitude elevation），非必传
     * @returns {string} return.identifier - 事件的唯一标识符，必传
     * @returns {number} return.timestamp - 上行数据传输时间。时间戳格式为数值，值为自GMT 1970年1月1日0时0分到当前时间点所经过的毫秒数，非必传
     * @returns {string} return.version - 版本号，默认default，非必传
     * @returns {Object} return.payload - 扩展数据，事件的输出参数（该数据根据产品Profile中定义的事件输出参数进行过滤，已存在定义的数据会保留，其余数据
     */
    if (inputData._bv_callback_message) {
        jsonMap['deviceId'] = inputData._bv_callback_message.deviceId;
        jsonMap['identifier'] = inputData._bv_callback_message.identifier;
        // 添加mask过滤
        var entityAttribute = CoreApi.getEntityAttribute(inputData._bv_callback_message.productId, inputData._bv_callback_message.deviceId);
        entityAttribute = JSON.parse(entityAttribute);
        if (typeof entityAttribute === 'object' && entityAttribute !== null && entityAttribute.hasOwnProperty('TestImageMask') && entityAttribute.hasOwnProperty('devi
            const mask = JSON.parse(entityAttribute.TestImageMask);
            inputData = maskFilter(inputData, mask)
            inputData.boundingBoxImage = drawBox(inputData, entityAttribute.deviceImage)
        }

        // 存在文件资源时将存储至网关服务下
        if (inputData.heatMapImage) {
            inputData.heatMapImageUrl = FileStorage.disk(inputData.heatMapImage, "jpg");
        }
        if (inputData.boundingBoxImage) {
            inputData.boundingBoxImageUrl = FileStorage.disk(inputData.boundingBoxImage, "jpg");
        }
        jsonMap['payload'] = {
            "analysisTime": inputData.analysisTime,
            "objectNumber": inputData.objectNumber,
            "objectBoundingBox": inputData.objectBoundingBox,
            "boundingBoxImageUrl": inputData.boundingBoxImageUrl,
            "heatMapImageUrl": inputData.heatMapImageUrl
        }
    }
    return jsonMap;
}
}

function drawBox(box,image) {
    const labels = [];
    const boxes = [];
    Object.keys(box.objectBoundingBox).forEach(function (key) {
        const value = box.objectBoundingBox[key];
        boxes.push(JtsGeometry.createCoordinatesXYXY(value));
        labels.push(key);
    });
    return JtsGeometry.drawBox(image, boxes, "Number: " + box.objectNumber, labels);
}

/**
 * 根据掩码（mask）过滤 box 对象中的边界框信息。
 * @param {object} inputData - 包含边界框信息的对象，其中 objectBoundingBox 属性存储具体的边界框数据。
 * @param {object} mask - 掩码对象，包含几何信息用于与边界框进行交集判断。
 */
function maskFilter(inputData, mask) {
    if (typeof inputData.objectBoundingBox !== 'object' && inputData.objectBoundingBox === null) {
        console.log("inputData.objectBoundingBox 不是有效的对象")
        return inputData;
    }
    const in_mask = mask.properties.mask_type === "positive";
    var objectBoundingBoxObj = {};
    var objectNumber = 0;
    for (const key in inputData.objectBoundingBox) {
        let value = inputData.objectBoundingBox[key];

        // value保留三位小数 double类型
        for (let i = 0; i < value.length; i++) {

```

```

    if (typeof value[i] !== 'number') {
        value[i] = parseFloat(value[i].toString());
    }
    value[i] = parseFloat(value[i].toFixed(3));
}

// 使用 JtsGeometry 工具类将边界框坐标转换为多边形所需的坐标数组
const boxCoord = JtsGeometry.createCoordinatesXYXY(value);
// 创建边界框对应的多边形
let boxPolygon = JtsGeometry.createPolygon(boxCoord);
for (let i = 0; i < mask.geometry.annotations.length; i++) {

    const maskCoord = JtsGeometry.createCoordinates(mask.geometry.annotations[i].coord);
    const maskPolygon = JtsGeometry.createPolygon(maskCoord);
    // 如果 in_mask 为 true 且边界框多边形和掩码多边形有交集
    if (in_mask && JtsGeometry.isIntersects(maskPolygon, boxPolygon)) {
        //console.log("box和mask有交集，交集面积为", JtsGeometry.intersectionArea(maskPolygon, boxPolygon))
        objectBoundingBoxObj[key] = value;
        objectNumber++;
        break;
    }
    // 如果 in_mask 为 false 且边界框多边形和掩码多边形没有交集
    if (!in_mask && !JtsGeometry.isIntersects(maskPolygon, boxPolygon)) {
        //console.log("负掩码，box和mask没有交集即为有效区域")
        objectBoundingBoxObj[key] = value;
        objectNumber++;
        break;
    }
}
}
inputData.objectBoundingBox = objectBoundingBoxObj;
inputData.objectNumber = objectNumber;
return inputData;
}

```

实体反控（下行）

```

/**
 * 将DTS平台的BV JSON数据转换为设备（或第三方服务）能识别的格式数据，DTS给设备（或第三方服务）下发数据时调用。
 * @param {JSONObject} serviceParams - 入参，请求时服务的固定参数
 * @param {string} serviceParams.requestId - 请求id
 * @param {string} serviceParams.productId - 请求产品编号
 * @param {string} serviceParams.deviceId - 请求设备编号
 * @param {string} serviceParams.serviceIdentifier - 请求服务标识
 * @param {string} serviceParams.serviceName - 请求服务名称
 * @param {JSONObject} inputParams - 入参，产品Profile中服务的输入参数
 * @param {number} flag - 入参，0=服务调用
 * @returns {JSONObject} - 出参，满足当前服务的输入参数，请注意数据类型以及参数标识符的统一性。
 */
function protocolToRawData(serviceParams, inputParams, flag) {
    if (flag === 0) {
        var data = inputParams;
        if (data.analysisType && data.analysisType.indexOf("100003") !== -1) {
            // 按分号分割 analysisMission 字符串
            var parts = data.analysisMission.split(';');
            var result = [];

            // 遍历每个分割后的部分
            for (var i = 0; i < parts.length; i++) {
                var item = parts[i].trim();
                if (item) {
                    var subParts = item.split(';');
                    var obj = {
                        deviceId: subParts[0].trim(),
                        programName: subParts[1].trim(),
                        originalVideoUrl: subParts[2].trim()
                    };
                    result.push(obj);
                }
            }
            data.analysisMission = [
                args: result,
                sceneGuid: "0"
            ];
            data.messageId = Date.now();
            return data;
        }
        if (data.beginTimestamp) {

```

```
// 获取事件发生时前5s的
data.beginTimestamp = new Date(data.beginTimestamp).getTime() - 5000;
return data;
}

//参考服务的输入参数
return inputParams;
}
}
```


常见问题与排障

附录

常见问题与故障排查

常见问题与故障排查 (FAQ)

本章整理了新用户在使用 DTS（数字孪生服务）接入和配置过程中，最常遇到的问题及详细排查步骤。

1. 属性上报成功，但实体详情中显示为“无数据”？

可能原因：

- **产品 Profile 未发布：**在创建/编辑产品 Profile（属性/事件/服务）后，必须点击右上角的“发布”按钮。未发布的 Profile 不会被平台的数据流网关识别。
- **标识符大小写不匹配：**设备上报的字段 Key（如 `temperature`）与产品 Profile 定义的属性标识符（如 `Temperature`）大小写不一致。DTS 的字段绑定是区分大小写的。
- **设备未关联网关：**检查网关详情中，是否已将该产品加入到“关联产品”列表中。

2. 网关自定义消息解析脚本执行时抛出 Sandbox Exception？

可能原因：

- **使用了禁用的语言特性：**DTS 消息解析脚本是在安全的沙箱隔离环境中执行的。脚本中禁止使用 Node.js 的原生 `require`、`fs` 模块、`eval`、以及多线程/进程控制语句。
- **全局变量冲突：**请确保所有局部变量皆使用 `var`、`let` 或 `const` 声明，禁止污染沙箱全局命名空间。

3. 配置的任务（规则引擎）没有触发？

排查步骤：

- **生效条件限制：**检查任务是否设置了生效时间段（如“生效条件 - 安全锁”），确保当前测试时间在生效窗口内。
- **属性更新触发模式：**DTS 任务属于“变动触发”。如果上报的属性值未发生改变（例如连续两次上报温度均为 `25` 且条件是 `> 25`），则不会重复触发执行动作。
- **查看执行日志：**在任务管理界面，点击当前任务的“执行日志”，查看具体的规则匹配判定过程。

产品性能限流说明

软件性能

产品与实体

限制项	描述	限制
标签Tag个数	单个产品、实体、网关或分组最多可以添加的标签数。	100
产品数量	单账号最多可以创建的产品数。	100
实体数量	单产品最多可以添加的实体数。	500,000
	单账号最多可以添加的实体数。	1,000,000
网关与实体	单个网关下最多添加的实体数。	1,500
产品Profile功能定义	单个产品最多可添加的功能数。	50
	struct类的属性，最多可添加的参数个数。	10
	当功能的数据类型为enum时，枚举项最多不超过30个。	30
	当功能的数据类型为text时，数据长度不超过2,048字节。	2,048字节
	当功能的数据类型为array时，数组内的元素不超过128个	128
	服务中可添加的入参和出参分别不超过20个。	20
	事件中可添加的出参不超过50个。	50
	导入产品模型profile时，文件大小不超过256 KB。	256 KB
数据解析	数据解析脚本文件大小不能超过48 KB。	48 KB
数据存储时间	产品运行时，产生的属性、事件、服务数据存储时间为30天。超出30天的数据不再保存。	30天

连接通信

限制项	描述	限制
网关接入限制	使用同一个网关证书信息（相同的productId、deviceId）。在同一时间，只能和BV数字孪生模块服务器建立一个连接多机部署时共享连接。	1
连接次数	每秒最大REDIS连接请求数。	500
	单网关每分钟最大连接请求次数。	5
网关订阅数	单网关的最大订阅数。超过订阅数的请求将会被直接拒绝。网关可以通过验证callbackURL消息，确认请求是否成功。	100
请求数量	每秒由网关向BV数字孪生模块发送的请求数。	1,000
	每秒由BV数字孪生模块向网关发送的请求数。	200
订阅关系限流	订阅关系，每秒向单应用程序发送的最大消息数。	1,000
消息通信限流	单网关每秒最多可上报的消息数量。	qos 0：30条/秒； qos 1：10条/秒； qos 2：10条/秒；
	单网关接收下行消息的最大限制为30条/秒，同时受限于网络环境。	30条/秒
消息缓存时长	qos0消息的最大缓存时间。qos为0时，丢弃发布请求消息。	不缓存
	qos1消息的最大缓存时间。qos为1时，丢弃发布请求消息。	不缓存
	qos2消息的最大缓存时间。如果最大时间后，未从接收端接收到callbackURL消息，则会丢弃发布请求消息。	根据重发次数设定，默认3次每10分钟一次。
消息重发机制	qos0消息的最大重发次数。	0次

限制项	描述	限制
	qos1消息的最大重发次数。 根据返回值确定是否重发， 无返回值时不做处理。 返回值的最大等待时间为10分钟。	0次
	qos2消息的最大重发次数。 根据返回值确定是否重发， 无返回值时不做处理。 返回值时重发每10分钟一次，最多重发3次。	3次
REDIS消息长度	REDIS单个发布消息最大长度。 超过此大小的发布请求将被直接拒绝。	256 KB
HTTP超时时间	响应HTTP请求的超时时间。	8秒

Topic相关

限制项	描述	限制
自定义Topic类数量	一个产品最多可以定义50个Topic类。	50
权限	实体只能对自己的Topic进行消息发布与订阅。	-
Topic长度	Topic长度不能超过128字节， UTF-8编码字符。	128字节
Topic类目	一个Topic中最多可包含多少个层级类目， 即Topic中斜杠的最大数量。	7
订阅数	每个订阅请求的最大订阅数。	8
操作生效时间	订阅和取消订阅都是操作10秒后生效， 一次订阅永久生效。 建议提前订阅Topic以免漏失信息。示例： 实体向TopicA发送SUB请求，10秒后，订阅生效， 实体开始收到实时消息，除非取消订阅， 实体将一直接收TopicA的消息。	10秒
广播Topic	一个广播Topic最多可以被1,000个接收端订阅。	1,000
	服务端SDK每秒只可发一条广播消息。	1条/秒