

目 录

数字孪生服务对接开发指南

概述

BV JSON格式说明

回执参数格式说明

实体接入API

HTTP协议接入(BV JSON数据格式)

设备上报

属性上报&事件上报

HTTP协议接入(透传/自定义)

设备上报

属性上报&事件上报

自定义Topic通信

自定义Topic上报

MQTT协议接入

CoAP协议接入

实体管理API

创建产品

获取产品列表

获取产品模型定义

获取产品详情

添加实体

获取实体列表

获取实体详情

修改实体详情

获取实体属性信息

删除实体

实体服务调用API

HTTP协议下行调用

自定义Topic下行调用

SDK下行调用

组织结构API

- 创建孪生体
- 添加实体到孪生体
- 删除孪生体
- 删除孪生体下实体
- 获取孪生体结构以及管理的实体

任务管理API

- 获取任务列表
- 执行任务
- 启动任务
- 停止任务
- 时序数据查询接口

获取订阅资源API

- 请求资源基础属性
- 请求资源所有属性
- 查询资源详细
- 批量查询资源详细
- 查询网关详细
- 批量查询网关详细
- 获取产品事件信息
- 请求应用程序或消费者加载的所有设备列表
- 批量查询产品的属性信息

订阅者接收订阅资源

自定义MCP工具说明

内置脚本二次开发参考

- Console类
- CoreApi类
- FileStorage类
- MockXMLHttpRequest类
- JtsGeometry类

数字孪生服务对接开发指南

数字孪生服务 (DTS) 对接开发指南

欢迎使用大视景数字孪生服务 (Digital Twin Service, 简称 DTS) 开放能力对接开发指南。本文档为第三方开发者提供全面的集成对接指导，帮助您高效地完成外部系统与 DTS 平台的双向数据流互通、设备管控及孪生体组织建模。

适用对象

根据开发职责的不同，本指南主要面向以下三类技术人员：

1. 物联网网关与设备端开发人员

- **关注焦点：**如何将物理设备或子系统通过网关接入 DTS、如何设计遥测上报报文、何时使用自定义解析脚本解析非标协议。
- **相关章节：**[实体接入 API](#)、[内置脚本二次开发参考](#)。

2. 第三方应用与系统集成开发人员

- **关注焦点：**如何向设备下发反控指令、如何动态创建或同步孪生产品与设备信息、如何被动接收 DTS 的实时数据推送、如何检索历史曲线。
- **相关章节：**[实体服务调用 API](#)、[实体管理 API](#)、[组织结构 API](#)、[任务管理 API](#)、[订阅者接收订阅资源](#)。

3. 孪生大屏与前端展示开发人员

- **关注焦点：**如何为低代码设计器或三维孪生面板拉取最新的网关和设备实时数据缓存。
- **相关章节：**[获取订阅资源 API](#)。

4. AI 智能体与大模型集成开发人员

- **关注焦点：**如何使智能体（如 CherryStudio、Claude Desktop）通过工具调用直接访问或反控数字孪生系统。
- **相关章节：**[自定义 MCP 工具说明](#)。

DTS 能做到的事情 (核心能力)

- **实体与本体模型管理：**通过 REST 接口动态定义产品模型（本体定义），并实现物理/虚拟实体的增、删、改、查与状态同步。

- **高并发双向数据通道**：提供标准的遥测上行上报通道和毫秒级响应的设备反控（下行服务调用）通道。
- **数字孪生空间与组织树**：支持动态构建孪生体组织树，支持将实体绑定到特定的地理空间节点（如：园区 -> 大楼 -> 楼层 -> 设备），为上层业务提供空间层级感知。
- **自动化任务与时序数据**：支持管理平台侧的后台联动规则任务，并提供时序数据库的高性能历史数据查询。
- **协议适配与边界脚本二次开发**：网关内置 JavaScript 执行沙箱，支持使用自定义脚本对非标准报文进行转换、就地清洗，并支持在图像上进行空间冲突检测与几何标记渲染。

🎯 API 场景选型决策指南 (何种场景选择什么样的 API)

为了方便开发者快速找到符合业务需求的 API 接口，我们整理了以下常用业务场景的 API 选型指南：

业务场景	对接方向	推荐调用的 API / 章节链接	选型与配置说明
设备实时遥测/事件上报	上行	🔗 实体接入 API	包含三种格式：标准/自定义/透传报文（需配合消息解析器）。MQTT Topic 接入。根据网关协议类型选择。
下发控制指令/反控物理设备	下行	🔗 实体服务调用 API	提供 HTTP 同步/异步、MQTT Topic 下发以及下行三种通道。支持配合回执参数使用。
管理孪生体层级空间树	双向	🔗 组织结构 API	用于创建孪生体（组织）、获取组织树以及向组织绑定或解绑设备实体。
手动执行或控制联动规则任务	双向	🔗 任务管理 API	动态启动、停止后台任务，或手动立即触发一次任务。
拉取实体的历史时序曲线	下行	🔗 任务管理 API - 时序数据查询	针对已绑定时序持续任务，按时间范围拉取历史数据，用于大屏曲线展示。

业务场景	对接方向	推荐调用的 API / 章节链接	选型与配置说明
大屏主动拉取设备/网关最新缓存	下行	🔗 获取订阅资源 API	包含请求单个/批量资源的最新属性、获取授权的网关及适合主动查询的拉
外部系统接收实时状态推送	上行	🔗 订阅者接收订阅资源	DTS 平台将数据实支持 HTTP 回调（消息队列（MQ）和适合被动接收的推
大模型与 AI 智能体集成	双向	🔗 自定义 MCP 工具说明	适合智能体（如 C Desktop）通过标准 Protocol (MCP) 进采用 Streamable S
编写自定义解析脚本	边缘/网关	🔗 内置脚本二次开发参考	在网关消息解析、调用内置基础类（CoreApi、FileStorMockXMLHttpRequestJtsGeometryUtils）

快速上手

环境准备

若您已部署好 DTS 服务（含私有化部署），或可直接访问公网 DTS 服务，无需额外准备环境，可直接进入后续对接流程。

若未部署相关服务，需先完成大视景数字孪生服务 DTS V2.0 及以上版本的部署：

- 1. **私有化部署流程：**请参考配套文档《大视景数字孪生服务 DTS 安装部署指南》，按指南完成部署配置；
- 2. **版本咨询：**关于 DTS 版本的适配要求、获取方式及部署相关疑问，可联系大视景科技官方对接人员咨询。

部署完成后，请确认 DTS 服务正常启动，可通过访问服务端口或测试接口验证服务可用性，确保环境满足对接要求。

核心对接使用场景架构图

场景一：数据层接入交互 (上行上报与下行控制)

适用场景：第三方应用需将自有实体数据接入 DTS 数据层，实现与 DTS 数字孪生数据的双向交互联动（如实体信息同步、遥测上报与反控）。

基于数据层接入交互，选择 [实体接入 API](#) 和 [实体服务调用 API](#) 了解更多。

场景二：主动获取数字孪生平台资源 (主动拉取模式)

适用场景：第三方应用需主动查询、获取 DTS 数字孪生平台中的各类资源最新状态（如孪生模型、实体最新遥测值等），按需拉取用于自身业务或大屏组件渲染。

获取数字孪生平台资源（主动模式），选择 [获取订阅资源 API](#) 了解更多。

场景三：被动订阅数字孪生平台资源 (实时推送模式)

适用场景：第三方应用无需主动查询，需要实时接收 DTS 数字孪生平台推送的数据变化、遥测更新、告警等信息，实现平台端与业务系统状态的实时同步。

订阅数字孪生平台资源（被动模式），选择 [订阅者接收订阅资源](#) 了解更多。

概述

概述

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 API可帮助您完成实体的接入注册和获取实体数据等功能。

使用DTS API 完成以下操作:

- [实体接入](#)
- [获取实体属性](#)
- [删除实体](#)
- [实体服务调用](#)
- [创建孪生体](#)
- [获取孪生体信息](#)
- [删除孪生体](#)
- [执行任务](#)
- [订阅资源流转](#)

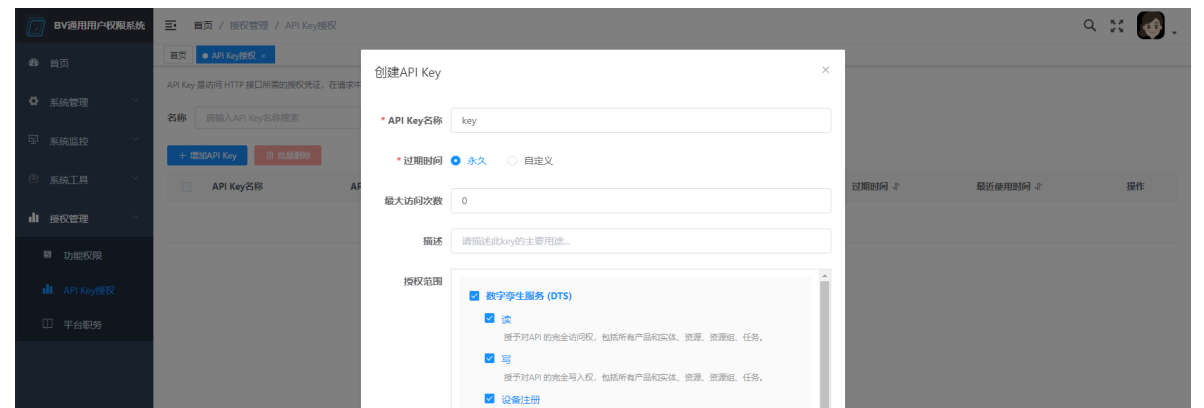
身份认证

为了对发送到DTS API 的请求进行身份验证，DTS 支持 HTTP 基本认证。请使用您的 API Key作为身份认证的密码。您可以通过 在通用用户系统中创建 API Key。

获取API Key

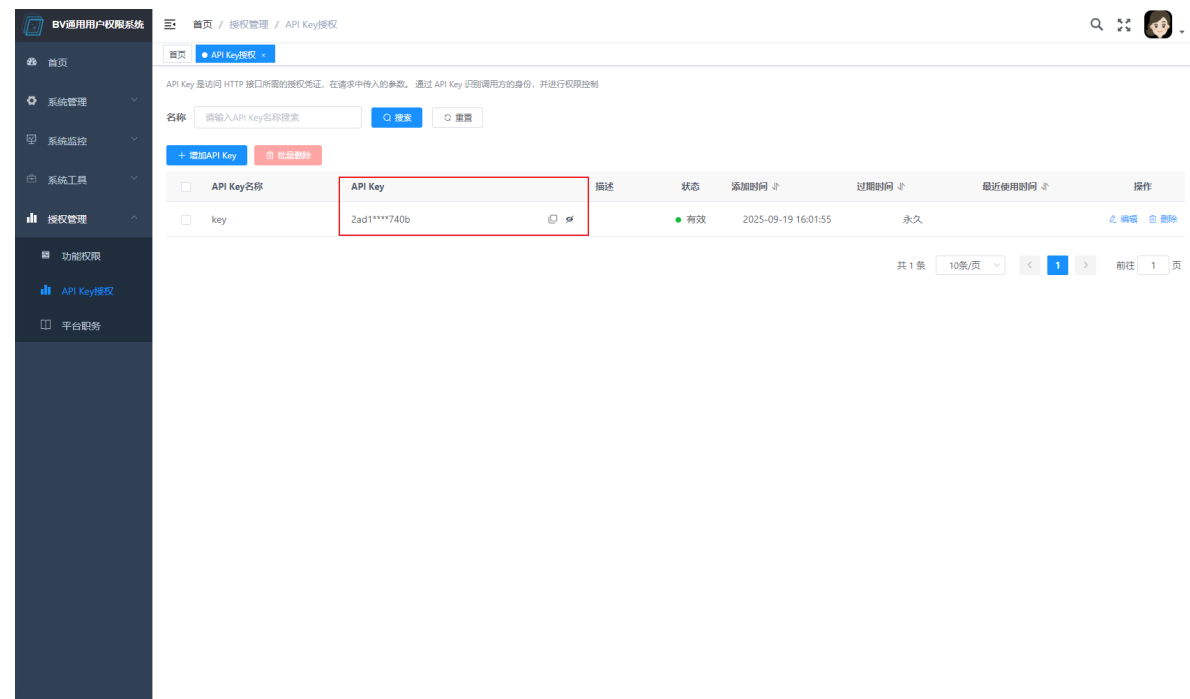
登录通用用户系统，进入“授权验证”模块。选择添加一个API Key，授予这个API Key与DTS相关的操作权限。生成的API Key需在进行请求HTTP接口调用时携带。

创建API Key





获取API Key



示例

```
curl --location --request POST 'http://Host/bv-microservice-digitaltwin-center/api/entity/attribute' \
--header 'User-Agent: Apifox/1.0.0 (https://apifox.com)' \
--header 'Content-Type: application/json' \
--header 'bv-api-key: YOUR_API_KEY' \
--header 'Accept: */*' \
```

获取订阅资源

1. 登录DTS，进入【订阅管理】模块，选择【订阅者列表】功能，进入订阅者中心；
2. 选择【创建订阅者】功能，手动输入或自动生成应用appSecret，输入订阅者名称，可选择输入订阅者描述文字，选择【确定】，创建订阅者；
3. 创建成功后，选择对应的订阅者，选择【查看】，进入订阅者详情页，获取appID（AK）和appSecret（SK）。

使用订阅资源流转相关API时需要。

API 规则

接口规则	相关描述
传输方式	支持HTTP传输
URL格式	URL格式：https://{hostname}:{port}/artemis/{uri}。hostname：平台的IP地址或域名。port：平台的web访问端口。URI：API的URI，每个API都有唯一的URI，详见API的接口定义。举例： https://127.0.0.1:443/artemis/api/xxxxx 注释：ip:port为平台web页面访问地址，
提交方式	采用Get/Post方法提交
数据格式	提交和返回数据都为JSON格式。自定义接入网关数据除外。
字符编码	统一采用UTF-8编码
安全认证	提供了基于API Kty的安全认证方案。只有认证通过，才能正常访问每一个API。
权限控制	每一个API都需要被授权。调用未被授权的API会提示错误。
返回值	见个接口返回值

BV JSON格式说明

BV JSON格式

BV JSON 是用于实体数据交互的统一标准格式。

实体注册

字段名称	字段类型	说明	是否
productId	string	产品的唯一标识符	是
deviceId	string	实体在产品下的唯一标识符	是
deviceName	string	实体名称	是
deviceStatus	number	实体状态 0未启用 1已启用（默认）	否
deviceType	number	实体类型0 动态资源（默认） 1 静态资源	否
location	string	实体位置经纬度（默认0 0 0 = latitude longitude elevation）	否
timestamp	number	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	否
version	string	版本号。若不传入此参数，则默认default	否
payload	object	上报的参数。说明事件上报时，为事件的输出参数；属性上报时为属性参数；	否

示例

```
{
  "productId": "ovw9GOI46TSqbedc",
  "deviceId": "d0ca3710-1575-4977-88a6-ee852082948b",
  "deviceName": "设备1739255549",
  "deviceStatus": 0,
  "payload": {
    "ipAddress": "164.27.227.218",
    "switchStatus": true,
    "brandInformation": "划见五选",
    "scheduledOnTime": "14:00:00",
    "scheduledOffTime": "14:32:29",
    "humidity": "0.13%",
    "temperature": "0.82°",
    "networkType": "IPv4",
    "connectionStatus": "已连接",
    "productModel": "面影好织世".
```

```
{
  "currentAndVoltage": {
    "realTimeVoltage": 220,
    "realTimeCurrent": 0.89
  },
  "powerRelated": {}
}
```

属性(事件)上报

字段名称	字段类型	说明	是否必
method	number	上行数据类型的标识。0属性上报；1事件上报；	是
productId	string	产品的唯一标识符	是
deviceId	string	实体在产品下的唯一标识符	是
deviceSecret	string	实体密钥与deviceId同时存在	否
deviceName	string	实体名称	否
deviceStatus	number	实体状态 0未启用 1已启用（默认）	否
identifier	string	事件唯一标识符（事件时必传）	否
location	string	实体位置经纬度（默认0 0 0 = latitude longitude elevation）	否
timestamp	number	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	否
version	string	版本号。若不传入此参数，则默认default	否
payload	object	上报的参数。说明事件上报时，为事件的输出参数；属性上报时为属性参数；	否

示例

```
{
  "method": 0,
  "productId": "oww9GOl46TSqbedc",
  "deviceId": "b28baf9c-40cc-4fe7-9488-8bc28f28717f",
  "deviceStatus": 0,
  "payload": {
    "ipAddress": "142.82.172.41",
    "switchStatus": false,
    "brandInformation": "出少素格",
    "scheduledOnTime": "14:00:00",
    "scheduledOffTime": "14:31:29",
    "humidity": "0.84%",
    "temperature": "0.43°",
    "networkType": "IPv4",
  }
}
```

BV JSON格式说明

```
{
  "connectionStatus": "已连接",
  "productModel": "技特国",
  "currentAndVoltage": {
    "realTimeVoltage": 220,
    "realTimeCurrent": 0.57
  },
  "powerRelated": {}
}
```

回执参数格式说明

回执参数格式说明

在数字孪生服务 (DTS) 中，服务调用与执行结果的流转遵循**指令下发-异步回传-脚本解析**的闭环控制逻辑。通用网关在下发服务调用时会携带回执相关参数，服务执行方在处理完毕后，需根据回执参数将结果上报回对应的回调地址（如 DTS 接收端），DTS 通过消息解析脚本识别对应的服务或事件标识符进行处理，从而形成完整的业务闭环。

核心工作流与应用场景

1. 业务流转过程

- 服务调用与指令下发：**用户或任务系统在 DTS 发起实体服务调用（例如：触发一次 AI 抓拍分析服务 `aiAnalysisOnce`）。DTS 在向服务执行方发送请求时，会随请求体下发回执参数 `_bv_callback_message`、回调方式 `_bv_callback_method` 和回调 URL `_bv_callback_url`。
- 任务处理与结果回调：**服务执行方根据入参进行业务处理。处理完成后，识别回执参数中的回调方式和地址：
- 如果回调方式是 `http` 或 `redis` 等，则将处理结果发送给 `_bv_callback_url` 中指定的地址（如果要回传给 DTS，该地址即为 DTS 接收地址）。
- 关键要求：在回调请求的数据体中，服务执行方必须**原样携带**下发时的 `_bv_callback_message` 回执对象。
- 数据上报与脚本解析：**DTS 网关接收到带有 `_bv_callback_message` 的回调数据后，利用产品的**消息解析脚本**进行解析。
- 解析识别与闭环：**解析脚本通过读取 `_bv_callback_message` 中携带的服务标识符（如 `aiAnalysisOnce`）或事件标识符（如 `aiEvent`），识别出这笔数据属于哪项服务或哪个事件的执行结果，完成报文清洗和模型字段填充，最终形成 DTS 指令发出与结果接收的闭环控制。

回执参数定义

1. 服务下行携带的控制回执字段（顶级字段）

字段名称	字段类型	说明	是否必传
<code>_bv_callback_message</code>	Object	回执核心元数据容器 (定义见下表)	是
<code>_bv_callback_method</code>	String	回调方式，例如 <code>http</code> 、 <code>redis</code>	是
<code>_bv_callback_url</code>	String	回调接收地址 (DTS通用网关的数据接收端点)	是

2. `_bv_callback_message` 结构详情

字段名称	字段类型	说明	是否必传
productId	String	产品唯一识别码	是
deviceId	String	实体设备唯一标识符	是
identifier	String	目标事件唯一标识符 (例如回调需要触发事件上报时存在)	否
_bv_service_callback	Object	服务调用元数据 (服务异步调用时存在， 定义见下表)	否

3. _bv_service_callback 结构详情

字段名称	字段类型	说明	是否必传
identifier	String	服务的唯一标识符 (例如: aiAnalysisOnce)	是
requestId	String	用于标记和匹配单次请求的唯一追踪 ID	是
version	String	产品模型版本，默认为 default	否
timestamp	Number	指令下发的时间戳（毫秒）	否

完整报文示例

以下是服务执行方（如 AI 算法分析服务器）在完成分析任务后，准备向 DTS 回调地址（_bv_callback_url）发送的 POST 请求报文：

```
{
  "_bv_callback_message": {
    "identifier": "aiEvent",
    "productId": "uhyVXMcYsdrNfucV",
    "deviceId": "D05A4AD9-0B97-0F4C-B6A9-96F84ABC7B3A",
    "_bv_service_callback": {
      "identifier": "aiAnalysisOnce",
      "requestId": "55c6ac78f8234bb18d23ffccfc5c0234",
      "version": "default",
      "timestamp": 1782112805338
    }
  },
  "includeImage": true,
  "image_data": "<base64 image: 165 chars>",
  "_bv_callback_method": "http",
  "analysisTypes": "person_vehicle",
  "_bv_callback_url": "http://66.100.100.66:40000/callback/returnDataToDts?productId=40014546&deviceId=..."
}
```

```
"_bv_callback_url": "http://192.168.100.38:48083/bv-universal-gateway/api/accessGateway/rawDataToProtocol/unyVXMCYSorNtUCV/L"
}
```

消息解析脚本（JavaScript）处理示例

当 DTS 接收到上述回调报文后，会将数据投入消息解析脚本（如 `rawDataToProtocol`）进行转换。在脚本中，开发者可以通过以下逻辑识别对应的服务与事件进行解析：

```
/**
 * DTS 消息解析脚本示例 (数据上行解析)
 * @param {string} rawData - 接收到的原始回调 JSON 字符串
 * @returns {string} - 解析清洗后的标准 BV JSON 字符串
 */
function rawDataToProtocol(rawData) {
    var payload = JSON.parse(rawData);
    var callbackMsg = payload._bv_callback_message;

    var result = {};

    // 1. 判断是否存在回执参数
    if (callbackMsg && callbackMsg._bv_service_callback) {
        var serviceId = callbackMsg._bv_service_callback.identifier; // 获取发起调用的服务标识符: "aiAnalysisOnce"
        var eventId = callbackMsg.identifier; // 获取目标事件标识符: "aiEvent"

        // 2. 根据服务标识符进行针对性的解析闭环
        if (serviceId === "aiAnalysisOnce") {
            result.productId = callbackMsg.productId;
            result.deviceId = callbackMsg.deviceId;
            result.method = 1; // 1 表示事件上报 (上报 AI 识别事件)
            result.identifier = eventId || "aiEvent";

            // 3. 将原始业务数据解析并组装到 payload 字段下
            result.payload = {
                analysisTypes: payload.analysisTypes,
                hasPersonOrVehicle: true, // 业务逻辑判断
                snapshotImage: payload.image_data // AI 人脸车牌图片数据
            };
        }
    }

    return JSON.stringify(result);
}
```

实体接入API

实体接入 API 选型与决策指南

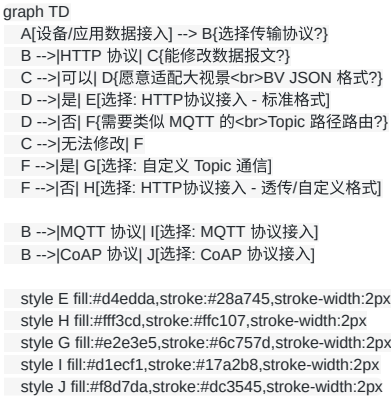
面向物联网设备与第三方应用的数据上报。为了确保数据能够正确地进入 DTS服务（大视景数字孪生服务）数据层，开发者需要根据设备端的技术特点和业务场景，选择最合适的接入方式。

实体接入方式对比

DTS服务通用网关提供了五种主要的数据上报接口，其对比与选型原则如下表所示：

接入方式	适用场景	报文格式要求
HTTP 协议接入 (标准 BV JSON 格式)	1. 设备或采集端具备灵活的报文组装能力。 2. 愿意在发送端直接适配大视景标准数据格式。	必须严格符合大视景标准的 BV JSON 格式规范。
HTTP 协议接入 (透传 / 自定义格式)	1. 硬件或采集端格式已固化，无法为 DTS服务修改数据结构。 2. 设备只能上报原始二进制、十六进制或第三方 JSON 报文。	任意格式（二进制 / 十六进制字节流 / 自定义 JSON 结构）。
自定义 Topic 通信 (自定义 Topic 上报)	1. 复杂的多租户、多产品统一物理网关路由场景。 2. 需要模拟 MQTT 的 Topic 路由机制，依据上报的 Topic 参数（如 <code>/{"productId}/{deviceId}/user/test</code> ）动态分发和匹配脚本。	自定义格式，但在请求体中必须携带 <code>topic</code> 字段以及业务载荷 <code>payload</code> 。
MQTT 协议接入	1. 采用标准的 MQTT 发布/订阅模式接入。 2. 第三方网关或设备作为 MQTT 客户端向 Broker发布数据。	建议为 JSON 格式。字段需与产品 Profile 或解析脚本一致。必须携带 <code>timestamp</code> 时间戳。
CoAP 协议接入	1. 受限设备（CPU/内存受限）采用 CoAP 轻量化协议上报。 2. 支持双向接入：设备上报（CoAP Server）与平台下行服务调用（CoAP Client）。	标准 JSON 格式。区分 <code>0=属性</code> 、 <code>1=事件</code> 、 <code>2=设备</code> 三种上报 URL 资源路径。

快速决策流程



接入方式详解及文档导航

HTTP 协议接入（标准 BV JSON 数据格式）

如果您的系统可以通过代码或配置，直接组装成 DTS服务要求的字段，请直接调用此组 API。* **开发参考：** * [设备注册/上报接口](#) —— 用于将实体注册到 DTS服务并上报基础状态。* [属性与事件上报接口](#) —— 用于设备遥测指标、属性及告警等事件的高频上报。* **数据规范：** 请先参阅 [BV JSON 格式说明](#)。

HTTP 协议接入（透传 / 自定义格式）

如果您希望在 DTS服务侧做数据适配，设备端只做最简数据透传，请直接调用此组 API。* **开发参考：** * [设备透传上报接口](#) —— 设备直接发送原始报文。* [属性与事件透传接口](#) —— 设备透传告警和遥测数据。* **脚本二次开发：** 由于需要编写转换逻辑，请参考 [内置脚本二次开发参考](#) 以及 [CoreApi 类库](#) 进行消息解析编写。

自定义 Topic 通信

如果您的应用类似于物联网网关转发器，并希望能通过特定 Topic 结构来动态查找设备和路由消息逻辑，请调用此接口。* **开发参考：** * [自定义 Topic 上报接口](#) —— 携带 Topic 标识的轻量化上报网关接口。

MQTT 协议接入

对于支持 MQTT 客户端的设备或第三方网关，可以通过发布/订阅消息的方式将数据同步到平台。* **开发参考：** * [MQTT 协议接入指南](#) —— 包含 Topic 规划、设备 ID 提取、设备注册、属性与事件上报说明及验收标准。

CoAP 协议接入

对于超低功耗或受限的物联网设备，可以使用轻量级 CoAP 协议（UDP）进行数据的上行上报及下行反控调用。* **开发参考：** * [CoAP 协议接入指南](#) —— 包含上/下行接入地址规范、属性/事件/设备上报协议规范及配置。

接入前置注意事项

1. **必须配置网关关联**：无论您选择哪种接入方式，创建的产品本体模型都**必须**在 DTS服务管理界面中关联到通用网关或您的自定义网关下，否则网关服务将无法接收对应 `productId` 的数据并会返回 `404` 或权限错误。
2. **凭证认证安全**：所有 HTTP 上行接入请求都需在请求头中携带 `bv-api-key: YOUR_API_KEY` 以进行安全鉴权，具体密钥可在用户系统的授权模块获取。详见 [身份认证概述](#)。

HTTP协议接入(BV JSON数据格式)

HTTP协议接入(BV JSON数据格式)

本文档详细介绍第三方应用采用HTTP协议、以BV JSON数据格式进行实体接入的API，明确接入要求、操作步骤及关键注意事项，助力开发者高效完成设备接入与数据上报对接。

适用场景

本接入方式适用于第三方应用采用HTTP协议发送**BV JSON格式**与DTS数字孪生系统进行**上行数据上报**的场景。

DTS系统提供标准HTTP协议接入通道，第三方应用需按要求将**BV JSON格式**的数据发送至指定HTTP协议接入服务。该服务将同步完成身份验证、请求参数识别、数据校验等核心操作，确保数据安全、准确接入。开发过程中，需严格遵循大视景科技提供的**HTTP协议实体接入（BV JSON数据格式）**对接规范执行。

注意事项

- 其他协议接入说明：**若需采用除HTTP协议外的其他协议进行实体接入，请联系大视景科技技术对接人员，由双方配合完成对应协议的网关适配与开发工作。
- API选择规范：**需严格按照产品配置的数据格式，选择对应的HTTP接入API，避免因API选择错误导致接入失败。
- 数据格式约束：**上报数据必须严格遵循BV JSON标准格式，字段缺失、格式错误将导致数据接入失败。
- 数据格式转换：**透传/自定义格式数据必须通过消息解析脚本转为BV JSON格式，确保DTS系统可正常解析。
- 标识符管理：**属性/事件的唯一标识符需提前规划、统一规范，避免标识符重复或缺失，影响请求识别与数据解析。
- 网关关联要求：**产品必须关联至DTS通用网关，否则无法完成数据传输与上报。

Apifox 接口分享：<https://apifox.com/apidoc/shared-2d55794d-55ae-43c8-b89a-3526536d2e35>

设备上报

设备上报

接口说明

向 DTS 平台注册实体，支持单实体/多实体批量上报；接口接收实体基础信息及自定义属性参数，并返回上报结果。

接口

项目	具体说明
请求方式	POST
接口路径	/bv-universal-gateway/api/accessGateway/deviceReport
访问地址	<code>http://{IP地址}:{端口}</code> 或 <code>http://{域名}:{端口}</code> （以实际部署为准）
Content-Type	application/json （必填）
字符编码	UTF-8

请求体

整体格式

请求体为 **JSON 数组**，数组元素为单个设备的上报信息对象，支持批量上报多个设备。

请求参数

字段名称	字段类型	是否必传	默认值	说明
productId	string	是	-	产品唯一标识符，由平台分配，不可为空
deviceId	string	是	-	设备在对应产品下的唯一标识符，不可为空
deviceName	string	是	-	设备名称，支持自定义，不可为空

字段名称	字段类型	是否必传	默认值	说明
deviceStatus	number	否	1	设备状态：0 = 未启用，1 = 已启用
deviceType	number	否	0	设备类型：0 = 动态资源，1 = 静态资源
location	string	否	“0 0 0”	设备位置经纬度，格式为 纬度 经度 海拔 （空格分隔，单位：度 米）
timestamp	number	否	当前时间戳	上行数据传输时间，GMT 1970-01-01 00:00:00 至今的毫秒数
version	string	否	“default”	版本号，支持自定义，例如 “v1.0.0”
payload	object	否	{}	上报的自定义参数：
- 事件上报： 填充事件输出参数				
- 属性上报： 填充设备属性参数				

请求示例

```
curl --location --request POST 'http://Host/bv-universal-gateway/api/accessGateway/deviceReport' \
--header 'User-Agent: Apifox/1.0.0 (https://apifox.com)' \
--header 'Content-Type: application/json' \
--header 'Accept: */*' \
--header 'Connection: keep-alive' \
--data-raw '{
  {
    "productId": "ovw9GOI46TSqbedc",
    "deviceId": "d0ca3710-1575-4977-88a6-ee852082948b",
    "deviceName": "设备1739255549",
    "deviceStatus": 0,
    "payload": {
      "ipAddress": "164.27.227.218",
      "switchStatus": true,
      "brandInformation": "划见五选",
      "scheduledOnTime": "14:00:00",
      "scheduledOffTime": "14:32:29",
      "humidity": "0.13",
      "temperature": "0.82",
      "networkType": "IPv4",
      "connectionStatus": "已连接",
      "productModel": "面影好织世",
      "currentAndVoltage": {
        "realTimeVoltage": 220,
        "realTimeCurrent": 0.89
      },
      "powerRelated": {}
    }
  }
}
```

响应说明

响应体格式

字段名称	字段类型	说明
code	number	响应状态码：0 = 成功，非 0 = 失败
message	string	响应描述：成功返回“success”，失败返回具体错误信息
info	object	扩展信息：成功时为 null，失败时可能返回参数错误明细等具体内容

响应状态码

code	message	排查建议
0	success	-
10000	未知错误	
10001	参数错误	校验参数类型与格式（如 timestamp 需为毫秒级数字、location 格式是否合规）
10024	缺少必传参数或参数内容为空	检查 productId、deviceId、deviceName 是否为空或未传递
20018	上报数据， 获取产品Profile失败， 请稍后重试	确认 productId 正确性， 检查产品是否已在平台完成创建,是否有发布
20020	上报数据失败， 获取产品Profile属性列表为空	确认目标产品已配置产品Profile
20022	实体上报失败， 实体数据列表为空	检查请求体是否为非空 JSON 数组， 禁止传递空数组或非数组格式
20023	上报数据失败， 获取产品数据解析脚本为空	确认是否配置产品的数据解析脚本

响应示例

设备上报

1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14. 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42. 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56. 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70. 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84. 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98. 99. 100.

成功响应

```
{
  "code": 0,
  "message": "success",
  "info": null
}
```

失败响应

```
{
  "code": 20018,
  "message": "上报数据失败，未找到产品Profile，请检查网关后重试",
  "info": null
}
```

属性上报&事件上报

属性上报&事件上报（BVJSON）

接口说明

通过**属性上报**或**事件上报**的方式，向 DTS 平台推送设备状态变化数据；支持两种上报类型按需选择，接口接收设备标识、上报类型及自定义参数，返回上报结果。

接口

| 项目 | 具体说明 |
|--------------|---|
| 请求方式 | POST |
| 接口路径 | /bv-universal-gateway/api/accessGateway/rawDataToProtocol |
| 访问地址 | <code>http://{IP地址}:{端口}</code> 或 <code>http://{域名}:{端口}</code> （以实际部署为准） |
| Content-Type | application/json（必填） |
| 字符编码 | UTF-8 |

请求体

整体格式

请求体为 **JSON 对象**，单条请求仅支持上报一种类型数据（属性/事件），不可同时上报两种类型。

请求参数

| 字段名称 | 字段类型 | 是否必传 | 默认值 | 说明 |
|-------------------------|--------|------|-----|-----------------------|
| method | number | 是 | - | 上行数据类型标识
（不可传其他值）： |
| 0 =
属性上报；1
= 事件上报 | | | | |

| 字段名称 | 字段类型 | 是否必传 | 默认值 | 说明 |
|--------------------------------|--------|------|-----------|--|
| productId | string | 是 | - | 产品唯一标识符，由平台分配，不可为空、不可传无效值 |
| deviceId | string | 是 | - | 设备在对应产品下的唯一标识符，不可为空、不可传无效值 |
| deviceSecret | string | 否 | - | 设备密钥，需与 deviceId 同时传入（按需使用，非必传） |
| deviceName | string | 否 | - | 设备名称，自定义填写（无需与注册时完全一致，仅作标识补充） |
| deviceStatus | number | 否 | 1 | 设备状态：0 = 未启用，1 = 已启用（默认值可无需手动传递） |
| identifier | string | 否 | - | 特殊说明： 事件上报（method=1）时必传 ，属性上报可忽略；为事件唯一标识符，需与产品事件模型配置一致 |
| location | string | 否 | "0 0 0" | 设备位置经纬度，格式为 纬度 经度 海拔 （空格分隔，单位：度米） |
| timestamp | number | 否 | 当前时间戳 | 上行数据传输时间，格式为GMT 1970-01-01 00:00:00 至今的毫秒数 |
| version | string | 否 | "default" | 版本号，自定义填写（如"v1.0.0"），不传递则使用默认值 |
| payload | object | 否 | {} | 上报的自定义参数（按需传递，非必传）： |
| - 属性上报：
设备属性参数
（如温度、电压等） | | | | |

| 字段名称 | 字段类型 | 是否必传 | 默认值 | 说明 |
|--------------------------------|------|------|-----|----|
| - 事件上报：
事件输出参数
(如告警相关数据) | | | | |

请求示例

属性上报示例（method=0）

```
// 属性上报
curl --location --request POST 'http://{访问地址}/bv-universal-gateway/api/accessGateway/rawDataToProtocol' \
--header 'User-Agent: Apifox/1.0.0 (https://apifox.com)' \
--header 'Content-Type: application/json' \
--header 'Accept: */*' \
--header 'Connection: keep-alive' \
--data-raw '{
  "method": 0,
  "productId": "ovw9GOI46TSqbedc",
  "deviceId": "b28baf9c-40cc-4fe7-9488-8bc28f28717f",
  "deviceStatus": 0,
  "payload": {
    "ipAddress": "142.82.172.41",
    "switchStatus": false,
    "brandInformation": "出少素格",
    "scheduledOnTime": "14:00:00",
    "scheduledOffTime": "14:31:29",
    "humidity": "0.84%",
    "temperature": "0.43°",
    "networkType": "IPv4",
    "connectionStatus": "已连接",
    "productModel": "技特国",
    "currentAndVoltage": {
      "realTimeVoltage": 220,
      "realTimeCurrent": 0.57
    },
    "powerRelated": {},
  }
}
```

事件上报示例（method=1）

```
// 事件上报
curl --location --request POST 'http://{访问地址}/bv-universal-gateway/api/accessGateway/rawDataToProtocol' \
--header 'User-Agent: Apifox/1.0.0 (https://apifox.com)' \
--header 'Content-Type: application/json' \
--header 'Accept: */*' \
--header 'Connection: keep-alive' \
--data-raw '{
  "method": 1,
  "productId": "ovw9GOI46TSqbedc",
  "deviceId": "b28baf9c-40cc-4fe7-9488-8bc28f28717f",
  "identifier": "PowerOverload",
  "deviceStatus": 0,
  "payload": {
    "ratedPower": 2313,
    "currentPower": 549
  }
}
```

响应说明

响应体格式

| 字段名称 | 字段类型 | 说明 |
|---------|--------|-----------------------------------|
| code | number | 响应状态码：0 = 成功，非 0 = 失败 |
| message | string | 响应描述：成功返回 “success”，失败返回具体错误信息 |
| info | object | 扩展信息：成功时为 null，失败时可能返回参数错误明细等具体内容 |

响应状态码

| code | message | 排查建议 |
|-------|----------------------------------|--|
| 0 | success | - |
| 10000 | 未知错误 | |
| 10001 | 参数错误 | 校验参数类型与格式（如 timestamp 需为毫秒级数字、location 格式是否合规） |
| 10024 | 缺少必传参数或参数内容为空 | 检查 productId、deviceId、deviceName 是否为空或未传递 |
| 20018 | 上报数据，
获取产品Profile失败，
请稍后重试 | 确认 productId 正确性，
检查产品是否已在平台完成创建,是否有发布 |
| 20020 | 上报数据失败，
获取产品Profile属性列表为空 | 确认目标产品已配置产品Profile |
| 20022 | 实体上报失败，
实体数据列表为空 | 检查请求体是否为非空 JSON 数组，
禁止传递空数组或非数组格式 |
| 20023 | 上报数据失败，
获取产品数据解析脚本为空 | 确认是否配置产品的数据解析脚本 |

响应示例

成功响应

属性上报&事件上报

```
{
  "code": 0,
  "message": "success",
  "info": null
}
```

失败响应

```
{
  "code": 20024,
  "message": "上行数据中的设备，与当前接入网关所关联的全部产品设备均不匹配，或该设备根本不存在。若您确定该网关关联的产品中确实存在此设备，请检查报",
  "info": null
}
```



HTTP协议接入(透传/自定义)

HTTP协议实体接入(透传/自定义)

本文档详细介绍第三方应用采用HTTP协议、以透传/自定义数据格式进行实体接入的所需的API，明确接入要求、操作步骤及关键注意事项，助力开发者高效完成设备接入与数据上报对接。

适用场景

此流程仅适用于第三方应用采用HTTP协议发送**透传/自定义** 格式与DTS数字孪生系统进行上行数据上报的场景。

本接入方式适用于第三方平台或设备通过HTTP协议，向DTS数字孪生系统上报上行数据的场景，上报数据可包括传感器采集数据、设备运行状态信息、环境监测数据等各类实体相关数据。上报时，第三方平台或设备无需提前转换数据格式，可直接将自身原生数据发送至DTS通用网关服务，该服务将同步完成数据解析、身份合法性验证、请求参数识别等核心操作，随后由DTS系统对接收的数据根据消息解析脚本进行进一步处理、存储，确保数据正常接入。

注意事项

- **其他协议接入说明：** 若需采用除HTTP协议外的其他协议进行实体接入，请联系大视景科技技术对接人员，由双方配合完成对应协议的网关适配与开发工作。
- **API选择规范：** 需严格按照产品配置的数据格式，选择对应的HTTP接入API，避免因API选择错误导致接入失败。
- **数据格式转换：** 透传/自定义格式数据必须通过消息解析脚本转为BV JSON格式，确保DTS系统可正常解析。
- **标识符管理：** 属性/事件的唯一标识符需提前规划、统一规范，避免标识符重复或缺失，影响请求识别与数据解析。
- **网关关联要求：** 产品必须关联至DTS通用网关，否则无法完成数据传输与上报。

Apifox 接口分享：<https://apifox.com/apidoc/shared-2d55794d-55ae-43c8-b89a-3526536d2e35>

设备上报

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

数组，设备集合

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-universal-gateway/api/accessGateway/deviceReport/{**productId**} |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-universal-gateway/api/accessGateway/deviceReport/{**productId**}" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

属性上报&事件上报

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

上报消息体

接口信息

| 项 | 内容 |
|--|--|
| 方法 | POST |
| 路径 | `/bv-universal-gateway/api/accessGateway/rawDataToProtocol/{productId}/{me |
| method
上行数据类型的标识。
0属性上报；
1事件上报；` | |
| 协议 | HTTP |

| | |
|--------|----|
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-universal-gateway/api/accessGateway/rawDataToProtocol/{**productId**}/{**method**}"
**method** 上行数据类型的标识。0属性上报；1事件上报；"\
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 bv-api-key |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

自定义Topic通信

自定义Topic通信

如果产品Profile功能不能满足您的业务需求，可以使用自定义Topic实现实体上下行通信。实体自定义Topic需在DTS从产品维度进行添加和管理，产品的自定义Topic类会自动映射为该产品下所有实体的自定义Topic。DTS对自定义Topic通信的数据格式没有要求，需自行自定义。本文介绍如何为产品自定义Topic类及自定义Topic的使用和通信说明。

适用场景

服务端下发消息到实体

服务端调用Pub，可向指定的自定义Topic发布消息。实体通过订阅该Topic，接收来自服务端的消息。

实体发送消息到服务端

实体向自定义Topic发布消息，服务端可通过物联网平台的AMQP订阅或云产品流转功能，接收来自实体的消息。

流程描述

1. 获取数字孪生服务（DTS）用户信息，登录DTS。
2. 创建产品，Topic类下添加自定义Topic。

自定义Topic上报

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

自定义上报消息体

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-universal-gateway/api/accessGateway/customReport |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-universal-gateway/api/accessGateway/customReport" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

MQTT协议接入

MQTT 第三方接入指南

面向第三方设备厂商、第三方平台开发人员及现场联调人员。本文档用于说明如何通过 MQTT 协议将终端设备接入 DTS服务，并对当前版本下上/下行控制能力的交互方式进行规范。

DTS服务（Digital Twin Service）提供了高性能的 MQTT 网关，支持通过发布/订阅模式实现高并发设备数据的双向联动。

在当前版本中，MQTT 协议网关的核心定位是**上行数据接入**：即设备或第三方平台作为客户端向 MQTT Broker 发布消息，DTS 服务的 MQTT 协议网关订阅对应 Topic，进而实现产品 Profile 解析和业务处理。

[!IMPORTANT] 当前版本的 DTS服务**不会主动向物理 Broker 发送控制指令 Topic**。如项目有下行反控需求，平台需通过 HTTP 接口、WebSocket、Redis 队列或 CoAP 客户端下发。设备在执行完毕后，可通过上行的 MQTT 属性或事件 Topic 将反控执行结果回传给 DTS服务。

对接目标

第三方完成接入时，双方需对齐以下数据与网络目标：

| 目标 | 第三方需要提供 | DTS服务处理方式 |
|---------|------------------------------|---|
| MQTT 连接 | Broker 物理地址、端口、账号、密码及 QoS 策略 | 平台配置网关客户端并建立持久连接订阅 |
| 设备注册 | 注册 Topic 及对应的 Payload 报文样例 | 提取设备信息完成动态自动建档注册 |
| 属性上报 | 遥测 Topic、载荷字段定义及样例数据 | 抽取关键遥测字段存入数字孪生实体属性 |
| 事件上报 | 告警/事件 Topic、事件编码及样例数据 | 写入孪生实体事件记录，触发报警规则联动 |
| 设备识别 | Topic 结构层级（如第几层），或消息体中的关键字段名 | 动态检索并提取设备物理 ID（ <code>deviceId</code> ） |
| 时间戳 | 消息生成的绝对时间戳（秒级或毫秒级） | 用于网络延迟校验及过期历史数据清洗拦截 |
| 下行控制 | 第三方接收反控指令的非 MQTT 地址通道 | 触发服务时由 DTS服务通过产品 Profile 接口直接调用 |
| | 目前上行接入 | |

| | | |
|----------|------------------------|-----------------------------------|
| 执行结果回传目标 | 属性上行 Topic 或事件上行 Topic | 监控执行回执并作为反控链路的最终闭环依据
DTS服务处理方式 |
|----------|------------------------|-----------------------------------|

当前能力边界

在接入调测前，请务必了解以下能力边界限制：

- MQTT 通道主要支持设备自动注册、属性遥测、事件告警。
- 每一个订阅的 Topic 必须能够明确地通过规则映射到**唯一的产品**和**唯一的上报类型**。
- 如果一条 MQTT 消息匹配不到路由规则，或者产生了冲突（同时命中多个产品或类型），DTS服务会直接将该消息丢弃。
- 同一产品模型定义下不允许配置重复的 MQTT 订阅 Topic。
- MQTT 消息具备时效性校验：平台默认会丢弃产生时间超过 300 秒的积压历史消息。
- 为满足多实例高可用部署，Broker 必须支持基于共享订阅模式（Shared Subscription）消费，例如消费路径为 `$share/{group}/{topic}`。
- MQTT 协议目前不能作为产品本体模型（Profile）的下行控制指令传输通道。

第三方需要提供的信息

MQTT Broker 信息

第三方需向平台提供以下 Broker 物理连接参数：

| 信息项目 | 样例配置 | 备注说明 |
|------------|------------------------------------|--------------------|
| Broker 地址 | <code>mqtt://127.0.0.1:1883</code> | 基础 TCP 传输地址 |
| 用户名 | <code>user001</code> | 若未启用鉴权可为空 |
| 密码 | <code>*****</code> | 若未启用鉴权可为空 |
| 推荐 QoS | <code>0</code> / <code>1</code> | 网关默认以 QoS 0 进行消费订阅 |
| 共享订阅支持 | 是 / 否 | 决定网关是否以多节点负载均衡形式消费 |
| Topic 订阅清单 | 详见 Topic 规则规划 | 每一条 Topic 的业务意义说明 |

若 Broker 物理环境不支持共享订阅，需在平台部署时避免运行多个网关实例重复订阅消费，以免造成数据乱序和性能浪费。

产品和设备信息

第三方提供数据对接方案时，需提前澄清：

- 本次 MQTT 网关接入包含哪些产品类型。
- 每个产品类型下挂载的具体设备实例 ID。

- 每一条 MQTT 消息报文中，如何定位该条消息所属的 `deviceId`（设备唯一 ID）。
- 自动注册设备模式下，注册报文里携带的设备中文名称等元数据映射规则。

Topic 规划规则

Topic 的语义结构应足够清晰，能让 DTS服务无歧义地识别消息归属的产品和上报数据分类。

推荐的黄金 Topic 设计规范为：“产品或设备类型 + 设备唯一标识 ID + 业务数据动作”，例如：

```
/device/{deviceId}/register/up  (设备注册)
/device/{deviceId}/property/up  (遥测上报)
/device/.../property/up        (事件上报)
```

若需对接第三方已有的非标 Topic 规范，需确保满足以下隔离原则：

- 自动注册、属性遥测、事件告警使用不同的 Topic 地址路径。
- 同一产品下，不要把不同业务逻辑的数据（例如属性和事件）发布到同一个 Topic。
- 避免不同产品的 Topic 匹配规则发生重叠或交叉覆盖。
- 如果设备 ID 嵌入在 Topic 中，请说明其所处的斜杠层级索引。
- 支持使用通配符（如 `+` 和 `#`），但通配符的使用应当保持克制，避免网关客户端大范围乱序消费。

Topic 范例参考：

| 业务类型 | Topic 示例 | 设备 ID 来源 |
|--------|------------------------------------|--------------------------------------|
| 设备自动注册 | <code>/device/+/register/up</code> | 提取自第 2 级 Topic 层级（ <code>+</code> 处） |
| 属性遥测上报 | <code>/device/+/property/up</code> | 提取自第 2 级 Topic 层级（ <code>+</code> 处） |
| 事件告警上报 | <code>/device/+event/up</code> | 提取自第 2 级 Topic 层级（ <code>+</code> 处） |

设备 ID 提取

DTS服务的 MQTT 网关支持两种设备 ID 提取策略，请在接入前与平台人员配置一致：

从 Topic 层级提取

如果设备将数据发布至以下 Topic：

```
/device/DEVICE_001/property/up
```

网关将其按 `/` 进行分割得到数组：

```
["", "device", "DEVICE_001", "property", "up"]
```

此时，物理 ID `DEVICE_001` 的层级位置索引为 `2`（从 0 开始计数）。

如果原始 Topic 开头不带斜杠：

```
device/DEVICE_001/property/up
```

分割后为：

```
["device", "DEVICE_001", "property", "up"]
```

此时设备 ID 处于层级索引 **1**。请注意开头斜杠对层级索引的影响。

从消息体字段提取

如果设备 ID 不在 Topic 中，也可以放在消息体顶层字段进行提取：

```
{
  "deviceId": "DEVICE_001",
  "timestamp": 1782377388128,
  "payload": {
    "temperature": 26.5
  }
}
```

[!IMPORTANT] 强烈建议将设备 ID 字段（如 `deviceId`）放置在 JSON 消息体的顶层结构中。放置在过深的多级嵌套字段内会导致网关引擎解析效率下降，增加路由延迟。

消息格式要求

MQTT Payload 必须采用标准的 UTF-8 编码的 JSON 格式。

此外，所有上行消息都必须携带时间戳属性：

- 支持秒级时间戳（例如 10 位整型）与毫秒级时间戳（例如 13 位整型）。
- 时间戳必须代表消息在终端设备上真正生成的时间，以便平台进行乱序过滤和延迟分析。
- 平台在路由解析时，会按如下优先级顺序检索消息体中的时间属性：
- 顶层 `timestamp`
- `data.timestamp`
- `payload.timestamp`

设备注册上报

设备注册上报用于向系统登记未激活的设备实体，以便在平台自动建档并绑定产品。

- Topic 示例：** `text /device/DEVICE_001/register/up`
- 单设备 Payload 样例：** `json { "deviceId": "DEVICE_001", "deviceName": "测试设备001", "timestamp": 1782377388128 }`
- 批量注册 Payload 样例：** `json [ { "deviceId": "DEVICE_001", "deviceName": "测试设备001", "timestamp": 1782377388128 }, { "deviceId": "DEVICE_002", "deviceName": "测试设备002", "timestamp": 1782377388128 } ]`

属性上报

属性上报用于高频更新数字孪生实体的属性指标（如温湿度、运行状态等）。

- **Topic 示例：** `text /device/DEVICE_001/property/up`
- **Payload 样例：** `json { "deviceId": "DEVICE_001", "timestamp": 1782377388128, "payload": { "onlineStatus": true, "batteryLevel": 86, "temperature": 26.5 } }` 注意：核心属性字段（如 `temperature`）是否嵌套在 `payload` 字段内取决于您配置的产品 Profile 本体映射方式。

事件上报

事件上报用于及时向系统报告发生的告警、故障或状态变更事件。

- **Topic 示例：** `text /device/DEVICE_001/event/up`
- **Payload 样例：** `json { "deviceId": "DEVICE_001", "timestamp": 1782377388128, "payload": { "eventCode": "lowBattery", "eventName": "低电量告警", "level": "warning", "batteryLevel": 18 } }` 注意：`eventCode` 代表事件编码，必须与平台定义的事件模型严格一致，以便后续触发联动控制规则。

下行控制协同

在当前版本设计中，DTS服务不通过 MQTT 向设备端发布下行指令，无法直接向如 `/device/DEVICE_001/command/down` 的 Topic 写入控制消息。

如果项目有设备反控需求，必须由第三方系统提供非 MQTT 协议的下行控制监听接口，例如：

- HTTP 接收接口（由网关发起 POST 访问）
- WebSocket 客户端通道
- RedisMQ 消息通道
- CoAP 协议 Server 接口
- MCP 协议服务端

DTS服务会在产品定义的服务动作中配置该非 MQTT 下行接口，并在触发反控逻辑时调用它。

推荐的反控闭环机制

推荐的上下行控制交互闭环链路如下：

```
DTS服务通过 HTTP/CoAP 等非 MQTT 协议调用第三方系统
-> 第三方系统接收并执行物理反控动作
-> 第三方系统将控制执行结果包装为属性或事件
-> 第三方通过 MQTT 上行通道发布至 Broker
-> DTS服务网关订阅获取执行回执并更新孪生实体状态
```

例如，DTS服务调用第三方 HTTP 开启设备的接口后，第三方在完成物理动作后，应主动向事件 Topic 发送一条事件消息：

```
{
  "deviceId": "DEVICE_001",
  "timestamp": 1782377388128,
  "payload": {
    "eventCode": "commandResult",
    "commandName": "start",
    "result": "success",
    "message": "执行成功"
  }
}
```

```
}  
}
```

如果第三方仅支持 MQTT 指令 Topic

如果受终端条件限制，第三方系统只能通过 MQTT 指令 Topic 接收反控控制，由于当前网关未包含此下行发布功能，需项目组另外进行定制化设计开发。此时需要额外定义：- 下行命令 Topic 结构与消息参数模板 - 下行命令 QoS 与超时应答策略 - 反控指令下发后的多实例 Broker 发布一致性设计 - 反控失败重试逻辑

联调步骤

联调前确认

接入调试前，第三方开发人员需先提供以下材料：

- MQTT Broker 物理连接信息（地址、端口、账户密码）。
- Broker 对共享订阅模式的支持情况。
- 完整的 Topic 订阅清单及消息 Payload 样例 JSON 文件。
- 设备物理 ID（`deviceId`）的提取位置（Topic 层级或 JSON 字段名）。
- 数据属性与事件映射对照表。
- 确认可用的非 MQTT 下行控制服务接口。

平台侧配置

由平台实施人员在后台完成以下配置：

1. 在 DTS服务管理控制台创建网关实例。
2. 配置 MQTT 网关连接客户端参数，使网关能成功连通物理 Broker。
3. 创建产品本体模型（Profile），配置消息解析脚本与参数映射规则。
4. 为产品绑定对应的 MQTT 消费路由规则。
5. 配置产品的下行反控服务（指向第三方的 HTTP / CoAP 服务）。

上行通信验证

第三方发送模拟报文进行联调：

1. 发布设备注册消息，检查 DTS服务控制台是否自动新增了该设备实例。
2. 发布属性遥测消息，在平台设备管理中查看属性值是否实时刷新。
3. 发布事件消息，验证平台事件日志中是否记录了该事件。
4. 连续发送多条数据，检验系统是否存在网络拥堵和数据丢失。

下行反控验证

若系统配置了下行逻辑，按以下步骤调试：

1. 平台管理员触发反控指令调用。
2. 第三方服务接收到下行 HTTP / CoAP 请求并执行物理控制。
3. 第三方执行完毕后，将结果消息组装并发送到约定好的 MQTT 上行 Topic。
4. 平台验证孪生状态或结果事件已被成功记录，闭环调试成功。

常见问题

消息发到 Broker 后平台没有数据

请依次排查：

- 检查 Topic 是否配置错误或大小写不匹配。
- 该 Topic 是否准确映射到唯一的 DTS服务产品路由。
- 消息载荷是否为严格的 JSON。
- 设备唯一 ID 是否按照预期的规则提取成功。
- 消息中的时间戳是否因网络延迟或偏差超过 300 秒而被系统过滤器丢弃。
- 检查 Broker 安全策略，确认网关客户端是否被授权订阅此 Topic。

为什么同一个 Topic 不能既做属性又做事件？

平台在消息路由阶段，必须先依据 Topic 快速区分数据分类以便送入不同的核心引擎处理。若同一个 Topic 被同时注册到属性和事件上，网关路由层将无法决定流转逻辑，会触发匹配冲突而抛弃消息。

`$share/{group}/{device}/+/{property}/up` 路径中为何有双斜杠？

若原始 Topic 路径以斜杠 `/` 开头，平台在开启共享订阅机制后，前置的 `$share/{group}/` 与 `/device/...` 拼接后就会自然产生 `//`。这是共享订阅规范的正常格式表现，无需担心。

为什么旧消息会被丢弃？

MQTT 网关主要面向实时设备监控。在网关网络中断恢复或网关重启期间，Broker 可能积压了大量的历史脏数据。如果这些积压的旧数据被不加甄别地重放，会覆盖平台的最新实时属性状态。因此网关会严格利用消息产生时间戳来清理超过 300 秒的历史积压消息。

可以不传 timestamp 吗？

强烈不建议。缺失时间戳会导致平台无法进行延迟计算、丢包排查和过期积压清理，这在正式对接场景中是不被接受的。

第三方交付清单

第三方完成接入准备后，需向项目组交付以下资料清单：

- MQTT Broker 的连接配置信息。
- shared subscription 的支持申明。
- 数据 Topic 清单及格式字段映射。
- 标准的设备自动注册、属性、事件 JSON 样例。
- 可用的反控下行 HTTP / CoAP 接口协议描述。
- 联调时间排班及技术联系人信息。

验收标准

MQTT 协议接入通过的验收标准如下：

- DTS服务网关客户端能连通 Broker 并建立稳定的订阅通道。
- 上行设备注册、遥测属性、告警事件能正确流转至 DTS服务关联的产品模型。
- 设备唯一识别 ID 提取完全正确，无设备实例数据错位。
- 消息投递处理延迟低于项目预定阈值，无明显消息积压。
- 过期脏数据消息被网关安全过滤器成功拦截。
- 反控链路可以通过非 MQTT 方式成功调用第三方执行器，且执行回执能通过 MQTT 上行通道成功闭环。

CoAP协议接入

CoAP 上下行对接指南

DTS服务（Digital Twin Service）提供了轻量化的 CoAP（Constrained Application Protocol）接入网关。本文档旨在帮助物联网设备端开发者与第三方系统集成商，快速理解并实现基于 CoAP 协议的双向数据互通。

协议说明

DTS服务的 CoAP 接入方案支持**上行数据采集**与**下行控制调用**两个方向的双向通信：

- 上行通信**：终端设备或第三方边缘系统作为 CoAP 客户端，主动向 DTS服务的 CoAP 服务端（Server）发起请求，上报属性遥测、事件告警或设备基础数据。
- 下行通信**：DTS服务作为 CoAP 客户端，主动向第三方部署的 CoAP 服务端（Server）下发请求，执行产品定义的服务调用或反控指令。

[!WARNING] 当前版本的 CoAP 网关通道不支持 DTLS 安全传输，亦不支持流式响应（Observe 订阅模式/分块传输）。通信采用标准非加密的 UDP 传输，报文最大建议在 MTU 限制（1280 字节）内，避免网络碎片化导致数据丢包。

上行接入地址

终端设备或第三方系统向上行上报数据时，请使用以下 CoAP 统一资源路径（URI）规范：

```
coap://{gatewayHost};5683/up/{productId}/{method}
```

参数详细说明如下：

| 参数 | 类型 | 说明 |
|-------------|---------|---|
| gatewayHost | String | 网关服务的主机地址或 IP 地址。 |
| productId | String | 在 DTS服务中定义的产品全局唯一标识符（Product ID）。 |
| method | Integer | 数据上报类型： <ul style="list-style-type: none">0：属性遥测上报（如温度、湿度等连续状态）1：事件告警上报（如设备开门、离线告警等突发状态）2：设备基本信息注册上报（向平台动态同步设备实例） |

属性上报

属性上报用于高频或定期同步设备实时的遥测指标及状态数据。

- **请求地址：** `text coap://{gatewayHost}:5683/up/{productId}/0`
- **请求方式：** POST
- **Payload 格式：** application/json
- **请求体示例：** `json { "deviceId": "device-001", "temperature": 4.5, "humidity": 62, "timestamp": 1782479141200 }` 字段说明：`deviceId` 为终端物理设备唯一 ID，`timestamp` 为毫秒级时间戳。其余字段（如 `temperature`）需与平台的产品 Profile 属性定义严格对齐。
- **返回示例：** `json { "code": 0, "message": "success" }` 返回码说明：`code: 0` 标识网关成功接收并投递至 DTS服务解析引擎。

事件上报

事件上报用于呈报设备在运行过程中产生的特定业务事件、系统故障或用户操作记录。

- **请求地址：** `text coap://{gatewayHost}:5683/up/{productId}/1`
- **请求方式：** POST
- **Payload 格式：** application/json
- **请求体示例：** `json { "deviceId": "device-001", "eventCode": "door_open", "eventTime": 1782479141200, "status": "open" }` 字段说明：`eventCode` 标识发生的事件类型编码，`eventTime` 为事件发生时的毫秒级时间戳，其余字段为该事件的输出参数。
- **返回示例：** `json { "code": 0, "message": "success" }`

设备上报

设备上报（设备注册）用于向平台动态宣告新设备实体的上线，以实现免建档设备自动入库。

- **请求地址：** `text coap://{gatewayHost}:5683/up/{productId}/2`
- **请求方式：** POST
- **Payload 格式：** application/json
- **请求体支持单设备对象或多设备数组。**
- **单设备请求体示例：** `json { "deviceId": "device-001", "deviceName": "测试设备001", "deviceStatus": 1, "timestamp": 1782479141200 }`
- **多设备批量请求体示例：** `json [ { "deviceId": "device-001", "deviceName": "测试设备001", "deviceStatus": 1 }, { "deviceId": "device-002", "deviceName": "测试设备002", "deviceStatus": 0 } ]`

上行数据格式说明

DTS服务网关层对 CoAP 上行 Payload 的数据格式支持以下两种处理机制：

1. **标准数据格式：**如果产品配置选择此模式，Payload 中字段名和数值必须与产品 Profile（本体模型）中定义的属性和事件标识符一一对应，网关将进行自动对齐解析。
2. **定制数据格式（透传模式）：**当设备端报文格式不可修改时，设备只需保证上报的是合法 JSON。网关层会将报文送入关联的产品 JavaScript 消息解析脚本中，由脚本完成字段转换与特征取值映射。

下行服务配置

当需要通过 DTS服务主动对物理设备进行控制或调用其服务时，网关提供了 CoAP Client 代理功能。

首先，需在网关管理控制台中，为目标网关服务实例配置 CoAP 服务：

| 配置项 | 说明 | 示例 |
|------|------------------------------|---------------------------------|
| 服务类型 | 协议类型 | CoAP |
| 服务地址 | 第三方 CoAP Server
接收请求的基准地址 | coap://127.0.0.1:19131/downlink |
| 请求方式 | 通信请求动作 | POST |

接着，在产品 Profile 定义的服务中，配置对应的动作路径。支持使用动态路由变量及参数占位符：

```
/qa/${route}?message=${message}
```

当触发服务调用时，网关会自动将 CoAP 服务基准地址与产品动作路径进行拼接，拼装出最终的 CoAP 请求路由：

```
coap://127.0.0.1:19131/downlink/qa/demo-route?message=hello
```

下行请求体

当下发控制指令时，DTS服务会将系统层面的服务调用参数组装为 CoAP Request Payload。

- 网关下发的 Payload 示例：`json { "route": "demo-route", "message": "hello" }`
- 第三方 CoAP 服务端处理完指令后返回的响应载荷必须是标准的 JSON 格式，网关会将其解析并作为服务调用回执返回给 DTS服务。

错误约定

网关在处理 CoAP 请求时，如果遇到以下非正常场景，会直接拦截并返回错误信息：

| 异常场景 | 错误处理结果 |
|---|-----------------------------|
| 请求体不是合法的 JSON 字符串 | 网关拒绝处理，返回 Bad Request 响应码 |
| 请求路径中的 <code>productId</code> 不存在或网关未加载该产品模型 | 网关拦截，返回 NotFound 错误 |
| 请求路径末尾的 <code>method</code> 参数不是 <code>0/1/2</code> | 网关拦截，返回 MethodNotAllowed 错误 |

| | |
|---------------------|--------------------|
| 异常场景
下行 CoAP | 错误处理结果
中心侧校验拦截， |
| 服务配置中返回流式响应或需要分块长响应 | 拒绝保存配置或执行反控调用 |

实体管理API

实体管理 API 总览

实体管理是大视景数字孪生服务 (DTS) 中的核心功能之一。通过这组 REST 接口，开发者可以在平台端动态定义产品模型（本体定义），并添加、修改、查询和删除具体关联的物理/虚拟设备实体。

功能说明

1. **产品模型定义**：

支持创建产品本体、定义标准功能（属性、事件、服务）、导出和查询产品模型。
2. **设备实例注册**：

在特定产品下，创建或删除单个/批量实体设备，生成接入所需的凭证。
3. **实体状态检索**：

提供产品和实体层级的列表查询、详情检索、属性数值获取和更新维护。

接口列表与导航

您可以通过点击下方链接直接跳转到具体接口的详细描述页面：

| 接口名称 | 调用方法 | 接口路径 | 核心作用 |
|---|------|--|--------------------|
|  创建产品 | POST | /bv-microservice-digitaltwin-center/product/create | 注册产品本体模型并定义其数据格式 |
|  获取产品列表 | POST | /bv-microservice-digitaltwin-center/product/findProductPage | 分页查询系统内产品 |
|  获取产品详情 | POST | /bv-microservice-digitaltwin-center/product/findProductDetail | 获取特定产品的详细数据 |
|  获取产品模型定义 | POST | /bv-microservice-digitaltwin-center/product/findProductProfile | 查询产品标准的属性、事件/服务定义 |
|  添加实体 | POST | /bv-microservice-digitaltwin-center/device/create | 在产品本体下注册设备并生成凭证 |
|  获取实体列表 | POST | /bv-microservice-digitaltwin-center/device/findDevicePage | 分页查询指定产品下的设备 |
|  获取实体详情 | POST | /bv-microservice-digitaltwin-center/device/findDeviceDetail | 获取特定实体的详细配置和网关代理信息 |
|  修改实体详情 | POST | /bv-microservice-digitaltwin-center/device/updateDeviceDetail | 更新设备名称、属性 |

| | | | |
|--|--------------|---|------------------|
|  修改实体详情
接口名称 | POST
调用方法 | center/device/update
接口路径 | 更新设备名称、1
核心作用 |
|  获取实体属性信息 | POST | /bv-microservice-digitaltwin-center/device/findDeviceProperties | 获取特定设备当前 |
|  删除实体 | POST | /bv-microservice-digitaltwin-center/device/delete | 从系统内移除指 |

创建产品

创建产品

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

调用该接口创建产品。

接口信息

| 项 | 内容 |
|------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/product/create |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |

| 项 | 内容 |
|--------|----|
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 |
|------------------------------|-------------------|------|-----------------------------------|
| addTime | string(date-time) | 否 | - |
| addUser | string | 否 | - |
| positionCoordinateSystemId | integer(int32) | 否 | 位置坐标系统编号 |
| productAccessGatewayProtocol | integer(int32) | 否 | 接入网关协议 0 标准网关协议 (HTTP) 1 自定义网关协议 |
| productAuthenticationMethod | integer(int32) | 否 | 认证方式 0 设备密钥 |
| productCategory | integer(int32) | 否 | 品类 0 自定义品类 1 标准品类 |
| productDataFormat | integer(int32) | 否 | 数据格式 0 透传/ 自定义 1 标准数据格式 (BV JSON) |
| productDescription | string | 否 | 产品描述 |
| productId | string | 否 | 产品编号 |
| productName | string | 否 | 产品名称 |
| productNormCategoryId | string | 否 | 标准品类编号 (选择自定义品类时为 null) |
| productSecret | string | 否 | 产品签名 |
| | | | 资源类型 0 动态资源 1 静态资源 |

| | | | |
|-------------------------|----------------------|-----------|---------------------------|
| productSourceType
字段 | integer(int32)
类型 | 否
是否必填 | 静态资源（poi） 2
说明
组合产品 |
| | | | |

| | | | |
|------------|-------------------|---|------------|
| templateId | string | 否 | 产品属性详情模板编号 |
| updateTime | string(date-time) | 否 | - |

请求体示例：

```
{
  "addTime": "",
  "addUser": "",
  "positionCoordinateSystemId": 0,
  "productAccessGatewayProtocol": 0,
  "productAuthenticationMethod": 0,
  "productCategory": 0,
  "productDataFormat": 0,
  "productDescription": "",
  "productId": "",
  "productName": "",
  "productNormCategoryId": "",
  "productSecret": "",
  "productSourceType": 0,
  "templateId": "",
  "updateTime": ""
}
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|---------|----------------|------|----|
| code | integer(int32) | 状态码 | 0 |
| info | object | - | {} |
| message | string | 响应消息 | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/product/create" \
-H "bv-api-key: YOUR_API_KEY" \

-H "Content-Type: application/json" \
-d '{"addTime":"","addUser":"","positionCoordinateSystemId": 0, "productAccessGatewayProtocol": 0, "productAuthenticationMethod": 0, "productCategory": 0, "p
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|--------------------|---|
| 200 | 200 | 创建成功
(返回产品详情) | 成功 |
| 201 | 201 | Created
(资源已创建) | 创建成功 |
| 401 | 401 | 未授权
(需要认证信息) | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | 禁止访问
(权限不足) | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | 接口不存在
(检查接口地址) | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取产品列表

获取产品列表

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

分页获取产品列表信息。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/product/findProductPage |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |

| 项 | 内容 |
|--------|----|
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------------|-------|------|--|---|
| currentPage | false | 否 | 当前页码
(默认第一页) | 0 |
| pageSize | false | 否 | 每页条数
(默认10条) | 0 |
| primaryIndustryCategory | false | 否 | 一级行业类别 | - |
| productName | false | 否 | 产品名称 | - |
| productSourceType | false | 否 | 类型 0 动态资源
1 静态资源
(poi) 2
组合产品 | 0 |
| productTagList | false | 否 | 产品标签 | [{"tagKey":
"",
"tagValue":
""}] |
| +tagKey | false | 否 | 标签key | - |
| +tagValue | false | 否 | 标签value | - |
| secondaryIndustryCategory | false | 否 | 二级行业类别 | - |
| sortColumn | true | 否 | 排序字段 0
产品名称 1
创建时间 2
更新时间 | 0 |
| sortOrder | true | 否 | 排序顺序 0 升序
1 降序 | 0 |

返回结果

获取产品列表

请求体示例：

```
{
  "currentPage": 0,
  "pageSize": 0,
  "primaryIndustryCategory": "",
  "productName": "",
  "productSourceType": 0,
  "productTagList": [
    {
      "tagKey": "",
      "tagValue": ""
    }
  ],
  "secondaryIndustryCategory": "",
  "sortColumn": 0,
  "sortOrder": 0
}
```

响应参数

| 字段 | 类型 | 说明 |
|------|----------------|----|
| code | integer(int32) | - |

| 字段 | 类型 | 说明 |
|------|-------------------|----|
| info | PageVO«ProductVO» | - |

| 字段 | 类型 | 说明 |
|-------------|----------------|--------|
| currentPage | integer(int64) | 当前页码 |
| | | |
| list | array | 分页数据列表 |
| | | |
| addTime | string | - |

| | | |
|---------------|--------------|------|
| addUser
字段 | string
类型 | 说明 |
| deviceCount | integer | 设备数量 |

| | | |
|------------------------------|-----------|--|
| gatewayVO | GatewayVO | 所属网关 |
| addTime | string | 创建时间 |
| bvProductList | array | 关联产品 |
| addTime | string | - |
| addUser | string | - |
| positionCoordinateSystemId | integer | 位置坐标系统编号 |
| productAccessGatewayProtocol | integer | 接入网关协议 0
标准网关协议
(HTTP) 1
自定义网关协议 |
| productAuthenticationMethod | integer | 认证方式 0 设备密钥 |
| productCategory | integer | 品类 0 自定义品类 1
标准品类 |
| productDataFormat | integer | 数据格式 0 透传/
自定义 1
标准数据格式 (BV
JSON) |
| productDescription | string | 产品描述 |
| productId | string | 产品编号 |
| productName | string | 产品名称 |
| productNormCategoryId | string | 标准品类编号
(选择自定义品类时为
null) |
| productSecret | string | 产品签名 |
| productSourceType | integer | 资源类型 0 动态资源 1
静态资源 (poi) 2
组合产品 |

| 字段 | 类型 | 说明 |
|------------|--------|------------|
| templateId | string | 产品属性详情模板编号 |

| | | |
|------------------------------|---------|------------------------------------|
| updateTime | string | - |
| deviceCount | integer | 设备数量 |
| gatewayAccessKey | string | 接入网关身份标识 |
| gatewayAccessSecret | string | 接入网关密钥 |
| gatewayDescription | string | 网关描述 |
| gatewayId | string | 编号 |
| gatewayIdentifier | string | 网关标识符 |
| gatewayName | string | 网关名称 |
| gatewayType | integer | 网关类型 0 Dumb网关（默认） 1 视频网关 2 第三方接入网关 |
| updateTime | string | 更新时间 |
| positionCoordinateSystemId | integer | 位置坐标系统编号 |
| primaryIndustryCategory | string | 一级行业类别 |
| productAccessGatewayProtocol | integer | 接入网关协议 0 标准网关协议（HTTP） 1 自定义网关协议 |
| productAuthenticationMethod | integer | 认证方式 0 设备密钥 |
| productCategory | integer | 品类 0 自定义品类 1 标准品类 |
| productDataFormat | integer | 数据格式 0 透传/自定义 1 标准数据格式（BV JSON） |
| productDescription | string | 产品描述 |

| 字段 | 类型 | 说明 |
|-----------|--------|------|
| productId | string | 产品编号 |

| | | |
|---------------------------|----------------|---------------------------------|
| productName | string | 产品名称 |
| productNormCategoryId | string | 标准品类编号
(选择自定义品类时为null) |
| productSecret | string | 产品签名 |
| productSourceType | integer | 资源类型 0 动态资源 1 静态资源 (poi) 2 组合产品 |
| secondaryIndustryCategory | string | 二级行业类别 |
| templateId | string | 产品属性详情模板编号 |
| updateTime | string | - |
| pageSize | integer(int64) | 每页条数 |
| totalPage | integer(int64) | 总页数 |
| totalRow | integer(int64) | 总记录数 |
| message | string | - |

响应示例：

```
{
  "code": 0,
  "info": {
    "currentPage": 0,
    "list": [
      {
        "addTime": "",
        "addUser": "",
        "deviceCount": 0,
        "gatewayVO": {
          "addTime": "",
          "bvProductList": [
            {
              "addTime": "",
              "addUser": "",
              "positionCoordinateSystemId": 0,
              "productAccessGatewayProtocol": 0,
              "productAuthenticationMethod": 0,
              "productCategory": 0,
              "productDataFormat": 0,
              "productDescription": "",
              "productId": "",
              "productName": ""
            }
          ]
        }
      }
    ]
  }
}
```

```
        "productNormCategoryId": "",
        "productSecret": "",
        "productSourceType": 0,
        "templateId": "",
        "updateTime": ""
    }
},
"deviceCount": 0,
"gatewayAccessKey": "",
"gatewayAccessSecret": "",
"gatewayDescription": "",
"gatewayId": "",
"gatewayIdentifier": "",
"gatewayName": "",
"gatewayType": 0,
"updateTime": ""
},
"positionCoordinateSystemId": 0,
"primaryIndustryCategory": "",
"productAccessGatewayProtocol": 0,
"productAuthenticationMethod": 0,
"productCategory": 0,
"productDataFormat": 0,
"productDescription": "",
"productId": "",
"productName": "",
"productNormCategoryId": "",
"productSecret": "",
"productSourceType": 0,
"secondaryIndustryCategory": "",
"templateId": "",
"updateTime": ""
}
},
"pageSize": 0,
"totalPage": 0,
"totalRow": 0
},
"message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/product/findProductPage" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"currentPage": 0, "pageSize": 0, "primaryIndustryCategory": "", "productName": "", "productSourceType": 0, "productTagList": [{"tagKey": "", "tagValue": ""}], "sec
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|------------------|------|
| 200 | 200 | 创建成功
(返回产品详情) | 成功 |

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

| | | | |
|------------|-------------------|--------------------------|---|
| 201
错误码 | 201
HTTP
状态 | Created
(资源已创建)
场景 | 创建成功
处理建议 |
| 401 | 401 | 未授权
(需要认证信息) | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | 禁止访问
(权限不足) | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | 接口不存在
(检查接口地址) | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取产品模型定义

获取产品模型定义

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取产品Profile信息。

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/product/getProfile |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|-----------|------|------|-----------|----|
| productId | true | 否 | productId | - |

| 字段 | 类型 | 说明 | 示例 |
|------|----------------|----|--|
| code | integer(int32) | - | 0 |
| info | ProductProfile | - | { "ev": "da", "ev": "ev", "": "ev", "ev": "ev", "ev": "0, ", [{"d": "typ", "ide": "ten", "pro": "0, ", "pro": [{"a": }, "da": "typ", "": "pro", "pro", "pro", "pro", "se", "ca |

| 字段 | 类型 | 说明 | 示例 |
|----|----|----|---|
| | | | "ga
"in
"int
"int
[{"
"se
""
"se
"tar |

| | | | |
|------------------------|---------|---------------------|---|
| events | array | - | <pre>{ "events": [{ "abstractId": "1", "dataDefinition": "data", "dataType": "string", "eventDescription": "event description", "eventId": "1", "eventIdentifier": "1", "eventName": "event", "eventPersistentStorage": 0, "eventQos": 1 }] }</pre> |
| abstractId | string | 摘要模板id | - |
| dataDefinition | object | 数据定义 | - |
| dataType | string | 数据类型 | - |
| eventDescription | string | 描述 | - |
| eventId | string | 事件id | - |
| eventIdentifier | string | 标识符 | - |
| eventName | string | 名称 | - |
| eventPersistentStorage | integer | 是否持久化存储 0 否 1 是 | - |
| eventQos | integer | 消息质量 0 默认 1 一级 2 二级 | - |

| | | | |
|---------------|---------------|----------------------------|----|
| eventRequired | integer | 是否为标准功能的必选 0 说明 1 必选 2 自定义 | 示例 |
| eventType | integer | 事件类型 0 消息 1 告警 | - |
| outputData | array | 输出参数 | - |
| dataTypeModel | DataTypeModel | - | - |
| specs | object | 类型约束定义 | - |

| | | | |
|------------------------------|----------------|---------|--|
| type | string | 数据类型 | - |
| description | string | - | - |
| identifier | string | - | - |
| name | string | - | - |
| templateId | string | 关联模板id | - |
| link | string | - | - |
| productAccessGatewayProtocol | integer(int32) | - | 0 |
| productDataFormat | integer(int32) | - | 0 |
| productId | string | - | - |
| properties | array | - | [{"a
{}, "
"da
"typ
""', '
"prc
"prc
"prc
"prc |
| addTime | string | 属性值更新时间 | - |
| dataDefinition | object | 数据定义 | - |
| dataType | string | 数据类型 | - |

| 数据类型 | string | 数据类型 | 示例 |
|---------------------|---------------|--------|----|
| 字段 | 类型 | 说明 | |
| dataTypeId | DataTypeModel | 数据类型对象 | - |
| specs | object | 类型约束定义 | - |
| type | string | 数据类型 | - |
| propertyDescription | string | 描述 | - |
| propertyId | string | 属性id | - |

| | | | |
|--------------------|-----------|---------------------------------------|--|
| propertyIdentifier | string | 标识符 | - |
| propertyName | string | 名称 | - |
| propertyReadWrite | integer | 读写类型 0 读写 1 只读 | - |
| propertyRequired | integer | 是否为标准功能的必选 0 可选 1 必选 2 自定义 | - |
| propertyValue | string | 属性值 | - |
| schema | string | - | - |
| services | array | - | [{"c": "ca", "ga": "in", "int": "int", "int": [{"}], "se": "", "se": "tar"}] |
| callbackMethod | integer | 回调方式 0 无 1 Http 2 Redis 3 属性上报 4 事件上报 | - |
| callbackUrl | Redis时必传) | 回调地址
(回调方式为Http | - |

| dataDefinition
字段 | object
类型 | 数据定义
说明 | 示例 |
|----------------------|---------------|------------|----|
| dataType | string | 数据类型 | - |
| gatewayEnvironmentId | string | 网关环境编号 | - |
| inputData | array | 输入参数 | - |
| dataTypeModel | DataTypeModel | - | - |
| specs | object | 类型约束定义 | - |

| | | | |
|-----------------------|---------------|-------------------------------|---|
| type | string | 数据类型 | - |
| description | string | - | - |
| identifier | string | - | - |
| name | string | - | - |
| interfaceResponseMode | integer | 第三方服务接口响应方式
0 同步 1 异步 2 流式 | - |
| interfaceUrl | string | 第三方服务接口地址 | - |
| outputData | array | 输出参数 | - |
| dataTypeModel | DataTypeModel | - | - |
| specs | object | 类型约束定义 | - |
| type | string | 数据类型 | - |
| description | string | - | - |
| identifier | string | - | - |
| name | string | - | - |
| serviceDescription | string | 描述 | - |
| serviceId | string | 服务编号 | - |
| serviceIdentifier | string | 服务标识符 | - |
| serviceName | string | 服务名称 | - |

| 字段 | 类型 | 是否为标准功能的必选 | 示例 |
|-----------------------|---------|----------------------------------|----|
| serviceRequired | integer | 可选 1 必选 2 自定义 | - |
| targetEventIdentifier | string | 目标事件标识符
(当回调方式为事件上报时存在，当前产品下) | - |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {
    "events": [
      {
        "abstractId": "",
        "dataDefinition": {},
        "dataType": "",
        "eventDescription": "",
        "eventId": "",
        "eventIdentifier": "",
        "eventName": "",
        "eventPersistentStorage": 0,
        "eventQos": 0,
        "eventRequired": 0,
        "eventType": 0,
        "outputData": [
          {
            "dataTypeModel": {
              "specs": {},
              "type": ""
            },
            "description": "",
            "identifier": "",
            "name": ""
          }
        ],
        "templateId": ""
      }
    ],
    "link": "",
    "productAccessGatewayProtocol": 0,
    "productDataFormat": 0,
    "productId": "",
    "properties": [
      {
        "addTime": "",
        "dataDefinition": {},
        "dataType": "",
        "dataTypeModel": {
          "specs": {},
          "type": ""
        },
        "propertyDescription": "",
        "propertyId": "",
        "propertyIdentifier": "",
        "propertyName": "",
        "propertyReadWrite": 0,
        "propertyRequired": 0,
        "propertyValue": ""
      }
    ],
    "schema": "",
    "services": [
      {
        "callbackMethod": 0,
        "callbackUrl": "",
        "dataDefinition": {},
        "dataType": "",
        "gatewayEnvironmentId": "",
        "inputData": [
          {}
        ]
      }
    ]
  }
}
```

```
    },
    "interfaceResponseMode": 0,
    "interfaceUrl": "",
    "outputData": [
      {}
    ],
    "serviceDescription": "",
    "serviceId": "",
    "serviceIdentifier": "",
    "serviceName": "",
    "serviceRequired": 0,
    "targetEventIdentifier": ""
  }
]
},
{
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/product/getProfile" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|-----------------------|---|
| 200 | 200 | 创建成功
(返回产品Profile) | 成功 |
| 201 | 201 | Created
(资源已创建) | 创建成功 |
| 401 | 401 | 未授权
(需要认证信息) | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | 禁止访问
(权限不足) | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | 接口不存在
(检查接口地址) | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取产品模型定义

获取产品详情

获取产品详情

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取产品详情信息。

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/product/findProductDetail |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体:

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|-----------|------|------|-----------|----|
| productId | true | 否 | productId | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|------|----------------|----|--|
| code | integer(int32) | - | 0 |
| info | ProductVO | - | {
"addTin": "
"deviceC"
{
"addTin"
[{
"addTir"
"position"
"product"
0, "prodi"
0, "prodi"
"product"
"product"
"product"
"product"
"product"
"product"
"templat"
""}], "dev"
"gatewa"
"gatewa"
"gatewa"
"gatewa"
"gatewa"
"gatewa"
"", "pos"
0, "primar"
"product"
0, "prod |

| 字段 | 类型 | 说明 | 示例 |
|----|----|----|---|
| | | | 0, "product": "product", "product": "product", "product": "product", "product": "product", "product": "product", "second": "second", "template": "template" |

| | | | |
|-------------|-------------------|------|---|
| addTime | string(date-time) | - | - |
| addUser | string | - | - |
| deviceCount | integer(int64) | 设备数量 | 0 |
| gatewayVO | GatewayVO | 所属网关 | { "addTin [{"addTir "position "product 0, "prodi 0, "prodi "product "product "product "product "product "product "product "templat ""}], "dev "gatewa "gatewa "gatewa "gatewa "gatewa "gatewa "gatewa ""} |
| addTime | string | 创建时间 | - |

| | | | |
|----------------------------|-------------|------------|---------|
| bvProductList
字段 | array
类型 | 关联产品
说明 | -
示例 |
| addTime | string | - | - |
| addUser | string | - | - |
| positionCoordinateSystemId | integer | 位置坐标系统编号 | 0 |

| | | | |
|------------------------------|---------|--|---|
| productAccessGatewayProtocol | integer | 接入网关协议 0
标准网关协议
(HTTP) 1
自定义网关协议 | 0 |
| productAuthenticationMethod | integer | 认证方式 0 设备密钥 | 0 |
| productCategory | integer | 品类 0 自定义品类 1
标准品类 | 0 |
| productDataFormat | integer | 数据格式 0 透传/
自定义 1
标准数据格式 (BV
JSON) | 0 |
| productDescription | string | 产品描述 | - |
| productId | string | 产品编号 | - |
| productName | string | 产品名称 | - |
| productNormCategoryId | string | 标准品类编号
(选择自定义品类时为
null) | - |
| productSecret | string | 产品签名 | - |
| productSourceType | integer | 资源类型 0 动态资源 1
静态资源 (poi) 2
组合产品 | 0 |
| templateId | string | 产品属性详情模板编号 | - |
| updateTime | string | | |

| updateTime
字段 | string
类型 | -
说明 | -
示例 |
|---------------------|--------------|----------|---------|
| deviceCount | integer | 设备数量 | 0 |
| gatewayAccessKey | string | 接入网关身份标识 | - |
| gatewayAccessSecret | string | 接入网关密钥 | - |
| gatewayDescription | string | 网关描述 | - |
| gatewayId | string | 编号 | - |

| | | | |
|------------------------------|----------------|--|---|
| gatewayIdentifier | string | 网关标识符 | - |
| gatewayName | string | 网关名称 | - |
| gatewayType | integer | 网关类型 0 Dumb网关
(默认) 1 视频网关 2
第三方接入网关 | - |
| updateTime | string | 更新时间 | - |
| positionCoordinateSystemId | integer(int32) | 位置坐标系统编号 | 0 |
| primaryIndustryCategory | string | 一级行业类别 | - |
| productAccessGatewayProtocol | integer(int32) | 接入网关协议 0
标准网关协议
(HTTP) 1
自定义网关协议 | 0 |
| productAuthenticationMethod | integer(int32) | 认证方式 0 设备密钥 | 0 |
| productCategory | integer(int32) | 品类 0 自定义品类 1
标准品类 | 0 |
| productDataFormat | integer(int32) | 数据格式 0 透传/
自定义 1
标准数据格式 (BV
JSON) | 0 |
| productDescription | string | 产品描述 | - |
| productId | string | 产品编号 | - |
| productName | string | 产品名称 | - |

| <code>productName</code>
字段 | <code>string</code>
类型 | 说明 | 示例 |
|------------------------------------|-----------------------------|---------------------------------|----|
| <code>productNormCategoryId</code> | <code>string</code> | 标准品类编号
(选择自定义品类时为 null) | - |
| <code>productSecret</code> | <code>string</code> | 产品签名 | - |
| <code>productSourceType</code> | <code>integer(int32)</code> | 资源类型 0 动态资源 1 静态资源 (poi) 2 组合产品 | 0 |

| | | | |
|--|--------------------------------|------------|---|
| <code>secondaryIndustryCategory</code> | <code>string</code> | 二级行业类别 | - |
| <code>templateId</code> | <code>string</code> | 产品属性详情模板编号 | - |
| <code>updateTime</code> | <code>string(date-time)</code> | - | - |
| <code>message</code> | <code>string</code> | - | - |

响应示例：

```
{
  "code": 0,
  "info": {
    "addTime": "",
    "addUser": "",
    "deviceCount": 0,
    "gatewayVO": {
      "addTime": "",
      "bvProductList": [
        {
          "addTime": "",
          "addUser": "",
          "positionCoordinateSystemId": 0,
          "productAccessGatewayProtocol": 0,
          "productAuthenticationMethod": 0,
          "productCategory": 0,
          "productDataFormat": 0,
          "productDescription": "",
          "productId": "",
          "productName": "",
          "productNormCategoryId": "",
          "productSecret": "",
          "productSourceType": 0,
          "templateId": "",
          "updateTime": ""
        }
      ],
      "deviceCount": 0,
      "gatewayAccessKey": "",
      "gatewayAccessSecret": "",
      "gatewayDescription": "",
      "gatewayId": "",
      "gatewayIdentifier": "",
      "gatewayName": "",
      "gatewayType": 0,
      "updateTime": ""
    },
    "positionCoordinateSystemId": 0,
    "primaryIndustryCategory": ""
  }
}
```

```
{
  "productAccessGatewayProtocol": 0,
  "productAuthenticationMethod": 0,
  "productCategory": 0,
  "productDataFormat": 0,
  "productDescription": "",
  "productId": "",
  "productName": "",
  "productNormCategoryId": "",
  "productSecret": "",
  "productSourceType": 0,
  "secondaryIndustryCategory": "",
  "templateId": "",
  "updateTime": ""
},
"message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/product/findProductDetail" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|--------------------|---|
| 200 | 200 | 创建成功
(返回产品详情) | 成功 |
| 201 | 201 | Created
(资源已创建) | 创建成功 |
| 401 | 401 | 未授权
(需要认证信息) | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | 禁止访问
(权限不足) | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | 接口不存在
(检查接口地址) | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

添加实体

添加实体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

添加一个实体（没有任何属性数据）到对应产品下。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/device/saveDevice |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |

| 项 | 内容 |
|--------|----|
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|-------------------|-------|------|---------------------------|----|
| addTime | false | 否 | 添加时间 | - |
| deviceDescription | false | 否 | 设备描述 | - |
| deviceGuid | false | 否 | 设备Guid（全局唯一） | - |
| deviceId | false | 否 | 设备id（产品下唯一） | - |
| deviceRemarkName | false | 否 | 设备名称 | - |
| deviceSecret | false | 否 | 设备签名
（和设备id组合使用） | - |
| deviceStatus | false | 否 | 设备状态 0 禁用 1 启用 | 0 |
| deviceType | false | 否 | 设备类型 0 设备 1 POI 2
组合设备 | 0 |
| location | false | 否 | 位置 | - |
| productId | false | 否 | 产品id | - |
| productName | false | 否 | 产品名称 | - |
| updateTime | false | 否 | 更新时间 | - |

请求体示例：

```
{
  "addTime": "",
  "deviceDescription": "",
  "deviceGuid": "",
  "deviceId": "",
  "deviceRemarkName": "",
  "deviceSecret": "",
  "deviceStatus": 0,
  "deviceType": 0,
  "location": "",
  "productId": "",
  "productName": "",
  "updateTime": ""
}
```

添加实体

```
updateTime :
}
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|---------|----------------|----|----|
| code | integer(int32) | - | 0 |
| info | object | - | {} |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/device/saveDevice" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"addTime": "", "deviceDescription": "", "deviceGuid": "", "deviceId": "", "deviceRemarkName": "", "deviceSecret": "", "deviceStatus": 0, "deviceType": 0, "location": ""}'
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|--------------------|---|
| 200 | 200 | 创建成功 | 成功 |
| 201 | 201 | Created
(资源已创建) | 创建成功 |
| 401 | 401 | 未授权
(需要认证信息) | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | 禁止访问
(权限不足) | 请检查该 API Key 是否被授权访问该接口 |

添加实体

| | | | |
|------------------------|--------------------------------|-------------------------|-----------------------|
| <div>404
错误码</div> | <div>404
HTTP
状态</div> | 接口不存在
场景
(检查接口地址) | 检查接口 URL 是否正确
处理建议 |
|------------------------|--------------------------------|-------------------------|-----------------------|

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取实体列表

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取指定产品下的所有实体（设备）列表，返回实体基本元数据列表（如设备ID、设备密钥、名称、状态、创建/更新时间以及位置等）。

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/api/entity/list |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/api/entity/list" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取实体详情

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取指定产品的指定实体（设备）详细信息，包括实体基本元数据（如名称、状态、密钥、唯一标识符）及其最新的动态上报属性（在 `payload` 字段中返回）。

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/api/entity/details |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/api/entity/details" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

修改实体详情

修改实体详情

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

修改指定实体的详情信息。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/device/updateDevice |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |

| 项 | 内容 |
|--------|----|
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|------------------------|-------|------|-----------------------------|----|
| equipmentInfo | true | 否 | equipmentInfo | - |
| addTime | false | 否 | 添加时间 | - |
| constituteItemParentId | false | 否 | 组合项父级id
(当设备类型为组合设备时存在) | - |
| detailLinkUrl | false | 否 | 详情链接地址 | - |
| deviceDescription | false | 否 | 设备描述 | - |
| deviceGuid | false | 否 | 设备Guid
(全局唯一) | - |
| deviceId | false | 否 | 设备id (产品下唯一) | - |
| deviceRemarkName | false | 否 | 设备名称 | - |
| deviceSecret | false | 否 | 设备签名
(和设备id组合使用) | - |
| deviceStatus | false | 否 | 设备状态 0 禁用 1 启用 | 0 |
| deviceType | false | 否 | 设备类型 0 设备 1 POI 2 组合设备 | 0 |
| gatewayId | false | 否 | 网关id | - |
| gatewayIdentifier | false | 否 | 网关标识符 | - |
| gatewayName | false | 否 | 网关名称 | - |
| gatewayType | false | 否 | 网关类型 0 Dumb网关 (默认) 1 视频网关 2 | 0 |

| | | | | |
|---------------------|-------------|-----------|-----------------------|--|
| gatewayType
字段 | false
类型 | 否
是否必填 | （默认）工业视频网关
第Ⅲ方接入网关 | 0
示例 |
| location | false | 否 | 位置 | - |
| productId | false | 否 | 产品id | - |
| productName | false | 否 | 产品名称 | - |
| properties | false | 否 | 属性 | [{"addTime":
"addUser": ""
"dataTypePa
"", "productIc
"productNam
"propertyDes
"", "propertyI
"propertyId
"", "propertyI
"",
"propertyRea
0,
"propertyRec
0, "propertyV
""}] |
| addTime | false | 否 | - | - |
| addUser | false | 否 | - | - |
| dataTypeParentId | false | 否 | 数据类型父级编号 | - |
| productId | false | 否 | 产品编号 | - |
| productName | false | 否 | 产品名称 | - |
| propertyDescription | false | 否 | 描述 | - |
| propertyId | false | 否 | 属性编号 | - |
| propertyIdentifier | false | 否 | 属性标识符 | - |
| propertyName | false | 否 | 属性名称 | - |
| propertyReadWrite | false | 否 | 属性读写类型 0 读写 1
只读 | - |

| 字段 | 类型 | 是否必填 | 只读说明 | 示例 |
|----|----|------|------|----|
|----|----|------|------|----|

| | | | | |
|-------------------------------|--------------------|---|-------------------------------|---|
| <code>propertyRequired</code> | <code>false</code> | 否 | 是否为标准功能的必选
0 可选 1 必选 2 自定义 | - |
| <code>propertyValue</code> | <code>false</code> | 否 | 属性值 | - |
| <code>updateTime</code> | <code>false</code> | 否 | 更新时间 | - |

请求体示例：

```
{
  "addTime": "",
  "constituteItemParentId": "",
  "detailLinkUrl": "",
  "deviceDescription": "",
  "deviceGuid": "",
  "deviceId": "",
  "deviceRemarkName": "",
  "deviceSecret": "",
  "deviceStatus": 0,
  "deviceType": 0,
  "gatewayId": "",
  "gatewayIdentifier": "",
  "gatewayName": "",
  "gatewayType": 0,
  "location": "",
  "productId": "",
  "productName": "",
  "properties": [
    {
      "addTime": "",
      "addUser": "",
      "dataTypeParentId": "",
      "productId": "",
      "productName": "",
      "propertyDescription": "",
      "propertyId": "",
      "propertyIdentifier": "",
      "propertyName": "",
      "propertyReadWrite": 0,
      "propertyRequired": 0,
      "propertyValue": ""
    }
  ],
  "updateTime": ""
}
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|-------------------|-----------------------------|----|-----------------|
| <code>code</code> | <code>integer(int32)</code> | - | <code>0</code> |
| <code>info</code> | <code>object</code> | - | <code>{}</code> |

| message
字段 | string
类型 | 说明 | 示例 |
|---------------|--------------|----|----|
|---------------|--------------|----|----|

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/device/updateDevice" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"addTime": "", "constituteItemParentId": "", "detailLinkUrl": "", "deviceDescription": "", "deviceGuid": "", "deviceId": "", "deviceRemarkName": "", "deviceSecret": ""}'
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|-------------------|---|
| 200 | 200 | OK (ResponseData) | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取实体属性信息

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取指定产品的指定实体（设备）当前的属性值信息（如系统电量、屏幕旋转状态、屏幕亮度等）。

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/api/entity/attribute |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/api/entity/attribute" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

删除实体

删除实体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

删除指定的实体。

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/device/delete |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|-------------|------|------|-------------|----|
| deviceGuids | true | 否 | deviceGuids | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|---------|----------------|----|----|
| code | integer(int32) | - | 0 |
| info | object | - | {} |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/device/delete" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|-------------------|------|
| 200 | 200 | OK (ResponseData) | 成功 |

| | | | |
|-----|------|--------------|---|
| 201 | 201 | Created | 创建成功 |
| 错误码 | HTTP | 场景 | 处理建议 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

实体服务调用API

实体服务调用 API 总览

实体服务调用是第三方应用向物理或虚拟实体下发控制指令（反控 / 下行调用）的统一通道。DTS 支持在产品模型（Profile）中定义服务，并通过本接口以多种协议下发执行。

调用方式选择

DTS 提供了三种下行服务调用通道：1. **HTTP 协议下行**：适用于直接使用标准 REST 接口反控直连设备的场景，网关会通过同步/异步方式返回执行状态。2. **自定义 Topic 下行**：适用于使用特定 MQTT Topic 订阅机制的场景，控制指令会封装为 Topic 载荷下发。3. **SDK 客户端下行**：为具备集成 SDK 能力的网关或复杂微服务提供底层高性能的反控下行通信。

接口列表与导航

您可以点击下方超链接跳转到对应的接口开发说明：

| 接口名称 | 调用协议/方式 | 接口路径 | 适用场景 |
|--|-------------|---|------|
|  HTTP协议下行调用 | HTTP | /bv-microservice-digitaltwin-center/api/service/invoke | 主机请求 |
|  自定义Topic下行调用 | MQTT / HTTP | /bv-universal-gateway/api/accessGateway/customDownlink | 自以格 |
|  SDK下行调用 | SDK / HTTP | /bv-microservice-digitaltwin-center/api/service/sdkInvoke | 集时 |

HTTP协议下行调用

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

服务调用消息体

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/service/call |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/service/call" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

自定义Topic下行调用

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

自定义下行调用消息体

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/topic/publish |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/topic/publish" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

SDK下行调用

订阅者调用产品Proile(服务)

订阅者（应用程序或服务端）通过openAPI（支持跨语言Thrift）方式进行鉴权与调用服务。订阅者调用服务前需要先进行鉴权获取Token令牌（[鉴权详细请参下行数据鉴权认证](#)），然后进行服务调用请求数字孪生参数为（ token 、 productId 、服务 identifier 、服务输入参数、 callbackURL ），数字孪生会进行数据重组转为网关交互标准格式以Topic方式与接入网关交互，接入网关则会使用 identifier 调用对应的方法，使用自定义协议方式与服务提供方交互，交互结果使用callbackURL返回。

协议格式

调用方式SDK

接入网关接收格式

| 参数 | 说明 |
|--------------|--|
| channel | topic/\${topic}。其中，变量\${topic}需替换为数据发往的目标Topic。 |
| requestId | 放在message中的参数，标记请求唯一主键。 |
| productId | 放在message中的参数，产品的Key，即BV数字孪生模块为产品颁发的全局唯一标识符。 |
| deviceId | 放在message中的参数，设备在产品下的唯一标识符。 |
| identifier | 放在message中的参数，服务唯一标识符。 |
| method | 放在message中的参数，功能名称。 |
| callType | 放在message中的参数，调用方式。 说明 0x00 异步 0x01 同步 |
| timestamp | 放在message中的参数，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。 |
| version | 放在message中的参数，版本号。若不传入此参数，则默认default。 |
| callbackType | 放在message中的参数，回调方式 redis或http |
| payload | 放在message中的参数，下发的参数。 说明 服务调用时表示输入参数，属性设置时表示属性的标识符和属性值。 |
| message | 发往\${topic}的数据内容。 |



组织结构API

组织结构 API 总览

在数字孪生世界中，设备实体通常不孤立存在，而是以层级、空间或业务结构组织在一起。组织结构 API 用于创建和管理这些业务层级关系（即**孪生体**树结构）。

核心概念：孪生体 (Twin)

- 孪生体可以是物理区域（如：大楼 ➡ 楼层 ➡ 房间）、也可以是逻辑组织（如：集团 ➡ 分公司 ➡ 车间）。
- 通过将具体的设备实体关联到指定的孪生体节点下，为孪生看板、低代码设计器及数据流分析提供空间感知的层级数据上下文。

接口列表与导航

您可以点击下方超链接跳转到对应的接口开发说明：

| 接口名称 | 调用方法 | 接口路径 | 核心作用 |
|--|------|--|--------------|
|  创建孪生体 | POST | /bv-microservice-digitaltwin-center/twin/create | 动态创建 |
|  添加实体到孪生体 | POST | /bv-microservice-digitaltwin-center/twin/addEntity | 将一个或 |
|  删除孪生体 | POST | /bv-microservice-digitaltwin-center/twin/delete | 移除整个 |
|  删除孪生体下实体 | POST | /bv-microservice-digitaltwin-center/twin/removeEntity | 解除具体 |
|  获取孪生体结构以及管理的实体 | POST | /bv-microservice-digitaltwin-center/twin/findTwinStructure | 查询并构
获取各节 |

创建孪生体

添加实体到孪生体

添加实体到孪生体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

指定实体添加到孪生体下

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-intelligent-spaces/api/intelligentSpaces/addEntityToTwin |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| | |

| | |
|-------------|----------|
| 超时时间建议
项 | 5s
内容 |
|-------------|----------|

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|------------------|-------|------|-------------------------|--|
| dto | true | 否 | dto | - |
| entityList | true | 否 | 实体列表 | [{"deviceId": "",
"deviceRemarkName":
"", "deviceSecret": "",
"productName": ""}] |
| deviceId | false | 否 | 设备id
(产品下唯一) | - |
| deviceRemarkName | false | 否 | 设备名称 | - |
| deviceSecret | false | 否 | 设备签名
(和设备id
组合使用) | - |
| productName | false | 否 | 产品名称 | - |
| productId | true | 否 | 产品编号 | - |
| twinId | true | 否 | 孪生体编号 | - |

请求体示例：

```
{
  "entityList": [
    {
      "deviceId": "",
      "deviceRemarkName": "",
      "deviceSecret": "",
      "productName": ""
    }
  ],
  "productId": "",
  "twinId": ""
}
```

响应参数

| 名称 | 类型 | 说明 | 示例 |
|----|----|----|----|
|----|----|----|----|

| 字段
字段 | 类型
类型 | 说明
说明 | 示例
示例 |
|----------|----------------|----------|----------|
| code | integer(int32) | - | 0 |
| info | object | - | {} |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-intelligent-spaces/api/intelligentSpaces/addEntityToTwin" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"entityList": [{"deviceId": "", "deviceRemarkName": "", "deviceSecret": "", "productName": ""}], "productId": "", "twinId": ""}'
```

接口错误码

| 错误码 | HTTP
状态 | 场景 | 处理建议 |
|-----|------------|----------------------|---|
| 200 | 200 | OK
(ResponseData) | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

添加实体到孪生体

删除孪生体

删除孪生体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

删除指定的实体。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-intelligent-spaces/intelligentSpaces/deleteIntelligentSpaces |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| | |

| | |
|-------------|----------|
| 超时时间建议
项 | 5s
内容 |
|-------------|----------|

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|--------------------------------------|--------------------|------|--|-----------------|
| <code>groupId</code> | <code>false</code> | 否 | 模板组编号
(删除单孪生体模板和结构孪生体模板的组下的一级孪生体时传) | - |
| <code>intelligentSpacesIdList</code> | <code>true</code> | 否 | 要删除的孪生体编号列表 | <code>[]</code> |

请求体示例：

```
{
  "groupId": "",
  "intelligentSpacesIdList": []
}
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|----------------------|-----------------------------|----|-----------------|
| <code>code</code> | <code>integer(int32)</code> | - | <code>0</code> |
| <code>info</code> | <code>object</code> | - | <code>{}</code> |
| <code>message</code> | <code>string</code> | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-intelligent-spaces/intelligentSpaces/deleteIntelligentSpaces" \
-H "bv-api-key: YOUR_API_KEY" \
```

```
-H "Content-Type: application/json" \  
-d '{"groupId": "", "intelligentSpacesIdList": []}'
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|-------------------|---|
| 200 | 200 | OK (ResponseData) | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

删除孪生体下实体

删除孪生体下实体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

删除指定的实体。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-intelligent-spaces/api/intelligentSpaces/deleteIntelligentSpacesResource |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| | |

| | |
|-------------|----------|
| 超时时间建议
项 | 5s
内容 |
|-------------|----------|

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------|-------|------|-----------------------|----|
| dtoList | true | 否 | dtoList | - |
| deviceId | false | 否 | 设备id | - |
| deviceSecret | false | 否 | 设备签名 | - |
| intelligentSpacesId | false | 否 | 设备所属空间id
(设备孪生体id) | - |

请求体示例：

```
[
  {
    "deviceId": "",
    "deviceSecret": "",
    "intelligentSpacesId": ""
  }
]
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|---------|----------------|----|----|
| code | integer(int32) | - | 0 |
| info | object | - | {} |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-intelligent-spaces/api/intelligentSpaces/deleteIntelligentSpacesResource" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"deviceId": "", "deviceSecret": "", "intelligentSpacesId": ""}'
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|-------------------|---|
| 200 | 200 | OK (ResponseData) | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取孪生体结构以及管理的实体

获取孪生体结构以及管理的实体

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

获取孪生体结构和它管理的实体。

接口信息

| 项 | 内容 |
|------|---|
| 方法 | POST |
| 路径 | /bv-microservice-intelligent-spaces/api/intelligentSpaces/deleteIntelligentSpacesResource |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| | |

| | |
|-------------|----------|
| 超时时间建议
项 | 5s
内容 |
|-------------|----------|

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------|-------|------|------------------|----|
| --- | --- | 否 | --- | - |
| intelligentSpacesId | true | 否 | 孪生体编号 | - |
| flag | false | 否 | 是否递归查询（0否 1是默认0） | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|---------|----------------|----|----|
| code | integer(int32) | - | 0 |
| info | object | - | {} |
| message | string | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-intelligent-spaces/api/intelligentSpaces/deleteIntelligentSpacesResource" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| | | | |
|--|--|--|--|
| | | | |
|--|--|--|--|

| 错误码
错误码 | HTTP
状态
状态 | 场景
场景 | 处理建议
处理建议 |
|------------|------------------|-----------------------------------|---|
| | | | |
| 200 | 200 | OK
(ResponseDataT«JSONObject») | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

任务管理API

任务管理 API 总览

任务是大视景平台内部运行的自动化业务规则（联动规则）。它允许开发者配置触发条件和动作，在后台自动执行。例如：当温湿度传感器上报温度超标时，触发空调实体降温服务，并将告警推送到钉钉。

主要能力

- 1. **任务控制**：获取自动化任务列表，手动控制各个后台规则任务的启动或停止，或直接触发一次性执行。
- 2. **时序历史查询**：对于绑定了时序持久化的实体属性，提供高性能的历史时序数据检索，用以支撑诊断分析。

接口列表与导航

您可以点击下方超链接跳转到对应的接口开发说明：

| 接口名称 | 调用方法 | 接口路径 | 核心作用 |
|--|------|---|----------------------------|
|  获取任务列表 | POST | /bv-microservice-digitaltwin-center/task/findTaskPage | 分页查询系统内配置的所 |
|  执行任务 | POST | /bv-microservice-digitaltwin-center/task/execute | 手动立即执行一次指定的1 |
|  启动任务 | POST | /bv-microservice-digitaltwin-center/task/start | 启动自动化任务，使能后i |
|  停止任务 | POST | /bv-microservice-digitaltwin-center/task/stop | 暂停自动化任务，禁用对l |
|  时序数据查询接口 | POST | /bv-microservice-digitaltwin-center/api/telemetry/query | 检索实体绑定的历史时序数
(如历史温湿度曲线) |

获取任务列表

获取任务列表

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

分页获取任务列表

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/sceneLinkage/rulePage |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|-------------|-------|------|-----------------------------|----|
| dto | true | 否 | dto | - |
| currentPage | false | 否 | 当前页码（默认第一页） | 0 |
| pageSize | false | 否 | 每页条数（默认10条） | 0 |
| ruleName | false | 否 | 规则名称 | - |
| sortColumn | true | 否 | 排序字段 0 场景名称 1 创建时间 2 最近执行时间 | 0 |
| sortOrder | true | 否 | 排序顺序 0 升序 1 降序 | 0 |

请求体示例：

```
{
  "currentPage": 0,
  "pageSize": 0,
  "ruleName": "",
  "sortColumn": 0,
  "sortOrder": 0
}
```

响应参数

| 字段 | 类型 | 说明 | 示例 |
|------|----------------------------|----|--|
| code | integer(int32) | - | 0 |
| info | PageVO«SceneLinkageRuleVO» | - | { "currentPage": 0, "pageSize": 0, "ruleName": "", "sortColumn": 0, "sortOrder": 0 } |

| 字段 | 类型 | 说明 | 示例 |
|-----------------------|----------------|--------------------------------------|---|
| currentPage | integer(int64) | 当前页码 | 0 |
| list | array | 分页数据列表 | [[{"addTi
"latestE
0,
"latestE
"", "ope
"ruleDe
"ruleId":
""}] |
| addTime | string | 创建时间 | - |
| latestExecutionStatus | integer | 最近执行状态
0 全部失败 1
全部成功 2
部分成功 | - |
| latestExecutionTime | string | 最近执行时间 | - |
| operatingStatus | integer | 运行状态 0
未启动 1
启动 | - |
| ruleDescription | string | 规则描述 | - |

| | | | |
|-----------|----------------|------|---|
| ruleId | string | 规则id | - |
| ruleName | string | 规则名称 | - |
| pageSize | integer(int64) | 每页条数 | 0 |
| totalPage | integer(int64) | 总页数 | 0 |
| totalRow | integer(int64) | 总记录数 | 0 |
| message | string | - | - |

响应示例：

```
{  
  "code": 0,
```

```
{
  "info": {
    "currentPage": 0,
    "list": [
      {
        "addTime": "",
        "latestExecutionStatus": 0,
        "latestExecutionTime": "",
        "operatingStatus": 0,
        "ruleDescription": "",
        "ruleId": "",
        "ruleName": ""
      }
    ],
    "pageSize": 0,
    "totalPage": 0,
    "totalRow": 0
  },
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/sceneLinkage/rulePage" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json" \
-d '{"currentPage": 0, "pageSize": 0, "ruleName": "", "sortColumn": 0, "sortOrder": 0}'
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---|--|
| 200 | 200 | OK
(ResponseDataT«PageVO«SceneLinkageRuleVO»») | 成功 |
| 201 | 201 | Created | 创建成功 |
| 401 | 401 | Unauthorized | 请登录通用用户系统并创建 API Key，并请求头中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取任务列表

执行任务

执行任务

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

手动触发一次执行，会跳过触发条件。

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/sceneLinkage/manualTrigger |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

| | |
|--|--|
| | |
|--|--|

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------|-------------------|------|------|----|
| <code>ruleId</code> | <code>true</code> | 否 | 规则id | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|----------------------|-----------------------------|----|-----------------|
| <code>code</code> | <code>integer(int32)</code> | - | <code>0</code> |
| <code>info</code> | <code>object</code> | - | <code>{}</code> |
| <code>message</code> | <code>string</code> | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/sceneLinkage/manualTrigger" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|------------------|------------------|----------------------|------|
| <code>200</code> | <code>200</code> | OK
(ResponseData) | 成功 |
| | | | |

| | | | |
|-----|------|--------------|---|
| 201 | 201 | Created | 创建成功 |
| 错误码 | HTTP | 场景 | 处理建议 |
| 401 | 状态 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| | 401 | | |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

启动任务

启动任务

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设置任务为启动状态

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/sceneLinkage/start |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------|-------------------|------|------|----|
| <code>ruleId</code> | <code>true</code> | 否 | 规则id | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|----------------------|-----------------------------|----|-----------------|
| <code>code</code> | <code>integer(int32)</code> | - | <code>0</code> |
| <code>info</code> | <code>object</code> | - | <code>{}</code> |
| <code>message</code> | <code>string</code> | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/sceneLinkage/start" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|------------------|------------------|----------------------|------|
| <code>200</code> | <code>200</code> | OK
(ResponseData) | 成功 |
| | | | |

| | | | |
|-----|------|--------------|---|
| 201 | 201 | Created | 创建成功 |
| 错误码 | HTTP | 场景 | 处理建议 |
| 401 | 状态 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| | 401 | | |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

停止任务

停止任务

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设置任务为停止

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/sceneLinkage/stop |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

| 字段 | 类型 | 是否必填 | 说明 | 示例 |
|---------------------|-------------------|------|------|----|
| <code>ruleId</code> | <code>true</code> | 否 | 规则id | - |

响应参数

| 字段 | 类型 | 说明 | 示例 |
|----------------------|-----------------------------|----|-----------------|
| <code>code</code> | <code>integer(int32)</code> | - | <code>0</code> |
| <code>info</code> | <code>object</code> | - | <code>{}</code> |
| <code>message</code> | <code>string</code> | - | - |

响应示例：

```
{
  "code": 0,
  "info": {},
  "message": ""
}
```

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/sceneLinkage/stop" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|------------------|------------------|----------------------|------|
| <code>200</code> | <code>200</code> | OK
(ResponseData) | 成功 |
| | | | |

| | | | |
|-----|------|--------------|---|
| 201 | 201 | Created | 创建成功 |
| 错误码 | HTTP | 场景 | 处理建议 |
| 401 | 状态 | Unauthorized | 请登录通用用户系统获取 API Key，并在 Header 中携带 <code>bv-api-key</code> |
| 403 | 403 | Forbidden | 请检查该 API Key 是否被授权访问该接口 |
| 404 | 404 | Not Found | 检查接口 URL 是否正确 |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

时序数据查询接口

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

上报消息体

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/sceneLinkage/bvSpaceAbstractTemplate/findSpaceAbstractResultS |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/sceneLinkage/bvSpaceAbstractTemplate/findSpaceAbstractResultSet" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取订阅资源API

获取订阅资源 API 总览

获取订阅资源 API 用于**数据消费者（第三方后台系统、前台看板等）**主动获取或批量同步平台产生的数据流。大视景内置了高效的数据流转过滤机制，确保消费者只拉取获授权订阅的范围。

主要功能

- **拉取遥测属性**：查询资源（实体）最新的单项或多项属性值。
- **同步网关与产品信息**：批量查询产品属性定义、网关详细及注册设备列表。
- **获取告警事件**：获取特定产品产生的下行或上行告警通知列表。

接口列表与导航

您可以点击下方超链接跳转到对应的接口开发说明：

| 接口名称 | 调用方法 | 接口路径 |
|--|------|---|
|  请求资源基础属性 | POST | /bv-microservice-digitaltwin-center/api/resource/baseProperties |
|  请求资源所有属性 | POST | /bv-microservice-digitaltwin-center/api/resource/allProperties |
|  查询资源详细 | POST | /bv-microservice-digitaltwin-center/api/resource/detail |
|  批量查询资源详细 | POST | /bv-microservice-digitaltwin-center/api/resource/batchDetail |
|  查询网关详细 | POST | /bv-microservice-digitaltwin-center/api/gateway/detail |
|  批量查询网关详细 | POST | /bv-microservice-digitaltwin-center/api/gateway/batchDetail |
|  获取产品事件信息 | POST | /bv-microservice-digitaltwin-center/api/product/events |

| | | |
|---|---------------------------------|--|
| <div>接口名称</div> <div>请求应用程序或消费者加载的所有设备列表</div> | <div>调用方法</div> <div>POST</div> | <div>接口路径</div> <div>/bv-microservice-digitaltwin-center/api/consumer/devices</div> |
| <div>批量查询产品的属性信息</div> | <div>调用方法</div> <div>POST</div> | <div>接口路径</div> <div>/bv-microservice-digitaltwin-center/api/product/batchProperti</div> |

请求资源基础属性

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/getBasicProperty |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/getBasicProperty" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

请求资源所有属性

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/getAllProperty |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/getAllProperty" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

查询资源详细

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/queryPropertyDetail |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/queryPropertyDetail" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

批量查询资源详细

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryPropertyDetail |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryPropertyDetail" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

查询网关详细

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/queryGatewayDetail |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/queryGatewayDetail" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

批量查询网关详细

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryGatewayDetail |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：

无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryGatewayDetail" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

获取产品事件信息

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/findEvent |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/findEvent" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

请求应用程序或消费者加载的所有设备列表

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|--|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/getAllEquipments |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/getAllEquipments" \
-H "bv-api-key: YOUR_API_KEY" \
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

批量查询产品的属性信息

文档信息

| 字段 | 内容 |
|------|---------------------------------|
| 关联产品 | [[DTS]] |
| 服务名称 | Microservice-DigitalTwin-Center |
| 接口类型 | 对外开放接口 |
| 协议类型 | HTTP |
| 当前版本 | v1.0.0 |
| 维护人 | 产品组 |
| 适用对象 | 后端开发 / 第三方集成商 |

服务接口简介

设备认证信息。JSON数据格式

接口信息

| 项 | 内容 |
|--------|---|
| 方法 | POST |
| 路径 | /bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryProductProperties |
| 协议 | HTTP |
| 是否鉴权 | 是 |
| 幂等性 | 否 |
| 超时时间建议 | 5s |

请求参数

请求体：
无请求参数。

响应参数

无响应参数。

调用示例

curl 示例：

```
curl -X POST "http://<Host>/bv-microservice-digitaltwin-center/openApi/resourceLoading/batchQueryProductProperties" \  
-H "bv-api-key: YOUR_API_KEY" \  
-H "Content-Type: application/json"
```

接口错误码

| 错误码 | HTTP 状态 | 场景 | 处理建议 |
|-----|---------|---------|-------------------------------|
| 400 | 400 | 参数错误或缺失 | 检查参数字段和必填项 |
| 401 | 401 | 未授权 | 检查请求头 <code>bv-api-key</code> |
| 403 | 403 | 无权限 | 检查接口授权 |
| 404 | 404 | 接口不存在 | 检查调用 URL |

备注

- 前置条件：API Key 已在“通用用户系统”中创建并关联了相应的 DTS 权限。
- 副作用：无。

订阅者接收订阅资源

订阅资源流转

建立订阅关系后，支持使用REDIS的方式进行数据上报；
订阅关系提供订阅BV数字孪生服务中所有资源消息，将资源数据或资源的一个或多个事件按指定的操作类型、订阅类型以及消息质量，将资源内产品下所有设备的上行数据（事件上报、属性上报、自定义上报）、资源流转（订阅资源），地理实体下所有POI的上行数据（属性上报）等流转至特定的服务端或应用平台或其下的某特定角色或用户。
订阅者指需要向服务端进行订阅的服务端或应用平台。消费者指在BV数字孪生平台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，数字孪生服务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。

准备事项

- 1.注册数字孪生服务DTService（以下简称DTS）账户
- 2.了解DTS的基本功能操作，可参考《[数字孪生服务DTService教程](#)》
- 3.已创建第三方服务端或应用平台的订阅者和订阅资源内容。可参考[订阅管理](#)

Redis 发布订阅

Redis 发布订阅(pub/sub)是一种消息通信模式：发送者(pub)发送消息，订阅者(sub)接收消息。Redis 客户端可以订阅任意数量的频道。

下图展示了频道 channel1，以及订阅这个频道的三个客户端—— client2、client5 和 client1 之间的关系：
当有新消息通过 PUBLISH 命令发送给频道 channel1 时，这个消息就会被发送给订阅它的三个客户端：

| 命令 | 描述 |
|---|----------------------|
| PSUBSCRIBE pattern [pattern ...] | 订阅一个或多个符合给定模式的频道 |
| PUBSUB subcommand [argument [argument ...]] | 查看订阅与发布系统状态 |
| PUBLISH channel message | 将信息发送到指定的频道 |
| PUNSUBSCRIBE [pattern [pattern ...]] | 退订所有给定模式的频道 |
| SUBSCRIBE channel [channel ...] | 订阅给定的一个或多个频道的信息 |
| UNSUBSCRIBE [channel [channel ...]] | 退订给定的频道 |
| PUBLISH channel* message | 将信息发送到所有channel开头的频道 |

流数据到应用程序

消息发送应用性

服务层将根据订阅配置进行异步数据处理，将消息发送到应用程序。

协议格式

| code | message |
|--------------|---|
| channel | /topic/\${topic}。其中，变量\${topic}需替换为数据发往的目标Topic |
| messageId | 放在message中的参数，消息体唯一主键 |
| productId | 放在message中的参数，产品的Key，
即BV数字孪生模块为产品颁发的全局唯一标识符 |
| deviceId | 放在message中的参数，设备在产品下的唯一标识符 |
| identifier | 放在message中的参数，属性或事件唯一标识符 |
| name | 放在message中的参数，功能名称 |
| type | 放在message中的参数，数据类型或事件类型 |
| qos | 放在message中的参数，消息质量。qos=0，当发生错误时，消息丢失；qos=1（暂不支持），当消息接收端宕机时，消息丢失；qos=2，保证消息质量。默认属性上报为qos=2 |
| timestamp | 放在message中的参数，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。 |
| version | 放在message中的参数，版本号。若不传入此参数，则默认default |
| templateId | 放在message中的参数，消息（事件、属性详情）模板唯一标识符 |
| templateURL | 放在message中的参数，消息（事件、属性详情）模板访问链接 |
| callbackURL | 放在message中的参数，消息回执接口 |
| payload | 放在message中的参数，上报的参数。说明：事件上报时，为事件的输出参数；属性上报时为属性参数。 |
| message-Type | 网关发送给BV数字孪生模块的上行数据的数据格式，目前支持JSON。 |
| message | 发往\${topic}的数据内容。 |

回执参数

| 回执参数 | 说明 |
|------------|----------------|
| code | 响应状态码。0：表示收到消息 |
| info | 消息内容 |
| +messageId | 消息体唯一主键 |
| +qos | 消息质量 |

```
{
  "code":0,
  "info":{
    "messageId":"854499b0df244a27",
    "qos":2
  }
}
```

自定义协议格式

数字孪生服务分发自定义Topic数据时，将使用数据透传（不会根据产品ProFile验证过滤属性）的方式进行数据传输。

自定义MCP工具说明

MCP 工具说明

更新时间：2026-06-30

本文件记录 DTS MCP 服务的连接方式和当前可用工具清单。工具清单通过 MCP `tools/list` 获取，服务端返回的服务信息为 `bv-ai-gateway`，版本 `1.0.0`。

基本信息

| 项目 | 说明 |
|--------|---|
| MCP 地址 | <code>http://192.168.100.56:18000/mcp</code> |
| MCP 类型 | Streamable HTTP |
| 协议版本 | 服务端协商版本为 <code>2025-03-26</code> |
| 鉴权方式 | 请求头 <code>bv-api-key: <YOUR_API_KEY></code> |
| 工具数量 | 36 |

真实 API Key 属于敏感凭证，文档和前端代码中建议只保留占位符，通过服务端配置、环境变量或密钥管理系统注入。

连接流程

MCP Streamable HTTP 客户端通常按以下顺序连接：

- 向 MCP 地址发送 `initialize` 请求。
- 从响应头读取 `Mcp-Session-Id`，后续请求都带上该会话 ID。
- 发送 `notifications/initialized` 通知。
- 调用 `tools/list` 获取工具列表。
- 调用 `tools/call` 执行具体工具。

通用请求头

Content-Type: application/json
Accept: application/json, text/event-stream
MCP-Protocol-Version: 2025-03-26
bv-api-key: <YOUR_API_KEY>
Mcp-Session-Id: <初始化响应返回的会话ID>

初始化请求第一次发送时不需要 `Mcp-Session-Id`。服务端可能返回普通 JSON，也可能返回 `text/event-stream`，客户端需要支持解析 SSE 中的 `data:` JSON 数据。

初始化示例

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "initialize",
  "params": {
    "protocolVersion": "2025-03-26",
    "capabilities": {},
    "clientInfo": {
      "name": "dts-mcp-client",
      "version": "1.0.0"
    }
  }
}
```

初始化成功后，服务端返回能力信息：

```
{
  "protocolVersion": "2025-03-26",
  "capabilities": {
    "completions": {},
    "logging": {},
    "prompts": {
      "listChanged": true
    },
    "resources": {
      "subscribe": false,
      "listChanged": true
    },
    "tools": {
      "listChanged": true
    }
  },
  "serverInfo": {
    "name": "bv-ai-gateway",
    "version": "1.0.0"
  }
}
```

获取工具列表示例

```
{
  "jsonrpc": "2.0",
  "id": 2,
  "method": "tools/list",
  "params": {}
}
```

如果响应中包含 `nextCursor`，需要继续带上该游标分页拉取。本次查询未返回 `nextCursor`。

调用工具示例

```
{
  "jsonrpc": "2.0",
  "id": 3,
  "method": "tools/call",
  "params": {
    "name": "devicePage",
    "arguments": {
      "currentPage": 1,
      "pageSize": 10,
      "productId": "产品编号"
    }
  }
}
```

所有工具的 `inputSchema` 均设置了 `additionalProperties: false`，调用时不要传入 schema 未声明的字段。

工具总览

| 分类 | 工具 |
|--------------------|---|
| 场景与大屏操作 | <code>switch_map</code> 、 <code>execute_label</code> 、 <code>execute_layout</code> 、 <code>execute_notification</code> |
| 项目、图层、节点、巡逻 | <code>get_the_project_list</code> 、 <code>find_layer_node_resource</code> 、 <code>nodePage</code> 、 <code>nodePageList</code> |
| 产品、实体、产品Profile、服务 | <code>productPage</code> 、 <code>productDetails</code> 、 <code>productFindFunctionList</code> 、 <code>productFindFunctionDetail</code> |
| 通知消息 | <code>find_notice_message</code> 、 <code>find_message_detail</code> 、 <code>send_comment</code> |
| 孪生体、任务、时序 | <code>twinsFindByName</code> 、 <code>twinsGet</code> 、 <code>taskPage</code> 、 <code>taskExecute</code> 、 <code>timeSequencePage</code> |
| 通用辅助 | <code>getCurrentDateTime</code> 、 <code>setAlarm</code> 、 <code>getTemperature</code> |

工具明细

场景与大屏操作

| 工具名 | 功能 | 必填参数 | 可选参数 |
|----------------------------|-----------|---|----------------------------------|
| <code>switch_map</code> | 切换地图、切换场景 | <code>name</code> string:
地图名称 | <code>AKA</code> string:
执行名称 |
| <code>execute_label</code> | 操作标签 | <code>executeAction</code> string:
执行动作，
可选值包括打开、关闭、执行、飞行、切换、跳转、定位
<code>name</code> string:
标签名称 | <code>AKA</code> string:
执行名称 |
| | | <code>executeAction</code> string: | |

| | | | |
|-----------------------|-------------------------|--|------------|
| 工具名
execute_layout | 功能
操作布局、面板、看板、
页面 | 执行参数，
可选值包括开启、
关闭
name string：
布局名称 | AKA
执行器 |
| execute_notifications | 操作通知中心 | executeAction
string：
执行动作，
可选值包括打开、
关闭 | AKA
执行器 |
| execute_highlight | 操作高亮 | executeAction
string：
执行动作，
可选值包括高亮、
取消高亮、
高亮所有、
取消高亮所有、
取消所有高亮
name string：
标签名称 | AKA
执行器 |

| | | | |
|----------------|------------|--|------------|
| execute_node | 操作地点、节点、位置 | executeAction
string：
执行动作，
可选值包括飞行、
切换、跳转、
定位、移动
name string：
地点名称 | AKA
执行器 |
| execute_patrol | 操作巡逻、轮巡、巡查 | executeAction
string：
执行动作，
可选值包括开启、
关闭
name string：
巡逻名称 | AKA
执行器 |
| | | intent string： | AKA |

| | | | |
|-----------------------------|---|--|--------------------------------|
| 工具名
executeCommand | 执行动作以及命令式操作，
功能
例如打开通知中心、
切换地图、开启轮巡等 | 必填参数
executeAction
string: 执行动作 | 可选参数
name
string: 执行对象名称 |
| filter_the_map_by_list_type | 按照列表或类型筛选地图 | flag integer: 0
按地图列表筛选，
1 按地图类型筛选
guid string:
项目编号 | name
string: 地图名称 |

项目、图层、节点、巡逻

| 工具名 | 功能 | 必填参数 | 可选参数 |
|--------------------------|---------------|-------------------------|---|
| get_the_project_list | 获取项目列表 | 无 | searchValue
string:
搜索内容 |
| find_layer_node_resource | 获取指定项目的图层节点资源 | guid
string:
项目编号 | layerType
integer:
图层类型
标签图层
矢量图层
模型图层
GIS 图层
特效图层
相机图层
单体化图
7 扩展 SI
图层
searchName
string:
搜索名称 |
| nodePage | 分页获取节点列表 | 无 | currentPage
integer:
当前页码
默认第一页
pageSize
integer:
每页条数
默认 10 条
nodeName
string: 节点名称 |

| 工具名 | 功能 | 必填参数 | 可选参数 |
|-----|----|------|---|
| | | | <div>string: 节点名称</div> <div>nodeType</div> <div>string: 节点类型</div> <div>projectG</div> <div>string: 项目编号</div> |

| | | | |
|------------------------------|--------------|--|--|
| <div>nodeGetInfoByName</div> | 根据名称获取节点信息 | <div>nodeName</div> <div>string: 节点名称</div> | <div>nodeType</div> <div>integer: 节点类型</div> <div>projectG</div> <div>string: 项目编号</div> |
| <div>nodeGetInfoById</div> | 根据 ID 获取节点信息 | <div>nodeId</div> <div>string: 节点 ID</div> | 无 |
| <div>findPatrolPath</div> | 获取巡逻路径 | <div>projectGuid</div> <div>string: 项目编号</div> | 无 |

产品、实体、产品Profile、服务

| 工具名 | 功能 | 必填参数 | 可选参数 |
|------------------------|----------|------|---|
| <div>productPage</div> | 分页获取产品列表 | 无 | <div>currentPage</div> <div>integer: 当前页码，默认第一页</div> <div>pageSize</div> <div>integer: 每页条数，默认 10 条</div> <div>productName</div> <div>string: 产品名称</div> |
| | | | |

| | | | |
|-------------------------|--------------------|----------------------------------|---|
| 工具名
productDetails | 获取产品详情 | 必填参数
productId
string：产品编号 | 可选参数 |
| | | | |
| productFindFunctionList | 获取产品的功能列表 | productId
string：产品编号 | functionType
integer：
功能类型，0
属性，1 服务
2 事件 |
| productGetProfile | 查看产品的产品
profile | productId
string：产品编号 | 无 |

| | | | |
|------------|----------|---|---|
| | | | currentPage
integer：
当前页码，
默认第一页 |
| | | | pageSize
integer：
每页条数，默
10 条 |
| devicePage | 分页获取实体列表 | 无 | productId
string：
产品编号 |
| | | | deviceId
string：
实体编号 |
| | | | deviceName |

| 工具名 | 功能 | 必填参数 | 可选参数 |
|--------------------|---|---|--|
| | | | 实体名称 |
| deviceDetails | 获取实体详情 | productId
string: 产品编号
deviceId
string: 实体编号 | 无 |
| deviceAttribute | 获取实体属性信息 | productId
string: 产品编号
deviceId
string: 实体编号 | 无 |
| deviceEventDataSet | 获取实体事件数据集，
可按消息、产品、
实体、
事件标识组合查询最新
事件信息 | 无 | messageId
string:
消息编号
productId
string:
产品编号
deviceId
string:
实体编号
eventId
string:
事件标识符 |

| | | | |
|-----------------------|--|---|---|
| productServiceExecute | 执行产品服务。
建议先查询服务输入参
数和服务标识符，
再查询实体与属性，
用属性值补齐服务入参 | productId
string: 产品编号
serviceIdentifier
string:
服务标识符
deviceId
string: 设备编号
inputData
object:
输入参数，例如
{ "identifier":
"123" } | 无 |
|-----------------------|--|---|---|

通知消息

| 工具名 | 功能 | 必填参数 | 可选参数 |
|----------------------------------|----------|---|--|
| <code>find_notice_message</code> | 获取通知消息列表 | <code>startTime</code>
string:
开始时间，格式
<code>yyyy-MM-dd</code>
<code>HH:mm:ss</code>
<code>endTime</code>
string:
结束时间，格式
<code>yyyy-MM-dd</code>
<code>HH:mm:ss</code>
<code>projectGuid</code>
string:
项目编号 | <code>productIdList</code> array:
产品 ID 列表
<code>deviceGuidList</code>
array: 设备 Guid
<code>intelligentSpaces</code>
array: 空间 ID 列表
<code>eventIdList</code> array:
事件 ID 列表
<code>messageType</code>
integer: 0 正常，
分享，2 上报，3 告警
<code>isRead</code> integer:
未读，1 已读，2 全部
<code>isComment</code> integer:
无评论，1 有评论
我评论，3 全部
<code>isWarn</code> integer:
一般消息，1
告警消息，2 全部
<code>isReply</code> integer:
不查看我回复，1
查看我回复
<code>isConfigTag</code> integer:
0 不查询无通知 Tag
的消息，1 查询无
Tag 的消息，2 全
<code>tagIdList</code> array:
tag 编号列表
<code>lastMessageTime</code>
string:
上一个消息的发生
时间
<code>searchValue</code> string:
搜索内容 |

| 工具名 | 功能 | 必填参数 | 可选参数 |
|----------------------------------|------------|--|--|
| <code>find_message_detail</code> | 获取通知消息详情 | <code>messageId</code>
string:
消息编号
<code>messageType</code>
integer:
消息类型, 0
正常, 1 分享,
2 上报
<code>projectGuid</code>
string:
项目编号 | <code>userId</code> integer:
用户编号
<code>operateMessageId</code>
string: 操作通知 ID |
| <code>send_comment</code> | 针对通知发送一条评论 | <code>messageId</code>
string: 消息 ID | <code>commentContent</code>
string: 评论内容 |

孪生体、任务、时序

| 工具名 | 功能 | 必填参数 |
|------------------------------|-------------|---|
| <code>twinsFindByName</code> | 根据名称模糊搜索孪生体 | <code>intelligentSpacesName</code>
string: 孪生体名称 |
| <code>twinsGet</code> | 获取孪生体及关联资源 | 无 |
| <code>taskPage</code> | 分页获取任务列表 | 无 |
| <code>taskExecute</code> | 执行任务, 只能执行 | <code>ruleId</code> string: 规则编号 |

| <code>taskExecute</code>
工具名 | <code>operatingStatus</code> 为 1 的任务
功能 | <code>templateId</code> string：规则编号
必填参数 |
|---------------------------------|---|---|
| <code>timeSeriesDataPage</code> | 分页获取可用的时序数据模板列表。
返回结果中的 <code>request</code> 字段可用于后续查询具体时序数据记录 | 无 |
| <code>timeSeriesDataGet</code> | 在没有明确请求参数时，
优先用于获取时序结果集 | <code>templateName</code> string：
时序名称 |

通用辅助

| 工具名 | 功能 | 必填参数 | 可选参数 |
|---------------------------------|----------------------|--|---|
| <code>getCurrentDateTime</code> | 获取用户时区的当前日期和时间 | 无 | 无 |
| <code>setAlarm</code> | 为指定时间设置用户提醒 | <code>time</code>
string：ISO-8601
格式时间 | 无 |
| <code>getTemperature</code> | 获取特定位置的温度，
单位为摄氏度 | 无 | <code>latitude</code>
number：
纬度
<code>longitude</code>
number：
经度 |

常见调用路径

查询并调用产品服务

- 用 `productPage` 或已知产品编号确定 `productId`。
 - 用 `productFindFunctionList` 查询服务列表，取服务标识符 `serviceIdentifier`。
 - 用 `devicePage` 查询实体，确定 `deviceId`。
 - 如需补齐入参，用 `deviceAttribute` 查询实体属性。
 - 用 `productServiceExecute` 执行服务。

查询通知并评论

1. 用 `find_notice_message` 按项目和时间范围查询通知列表。
2. 用 `find_message_detail` 查看通知详情。
3. 用 `send_comment` 对通知发送评论。

控制大屏场景

1. 用 `get the project list` 确定项目。
2. 用 `switch_map` 切换地图或场景。
3. 用 `execute_label`、`execute node`、`execute_layout`、`execute_patrol` 等工具执行具体动作。
4. 复杂自然语言命令可统一走 `executeCommand`，由服务端按 `intent`、`executeAction`、`commandParams` 解释执行。

对接注意事项

- 所有时间字段建议使用服务端要求的格式；通知消息列表中的 `startTime`、`endTime` 为 `yyyy-MM-dd HH:mm:ss`。
- 分页接口未传 `currentPage`、`pageSize` 时，按工具说明使用默认第一页和默认 10 条。
- 枚举值请按工具描述传入，避免传中文说明文本代替枚举数字。
- `inputData`、`commandParams` 等对象型参数应传 JSON 对象，不要传 JSON 字符串。
- 执行型工具会改变大屏、任务或设备服务状态，生产环境接入前建议先做权限隔离和操作确认。

内置脚本二次开发参考

基础工具类函数

本文档目录总结了各个类的接口信息，包括类作用、主要方法列表。详细内容请跳转到对应的文档查看。

ConsoleJS

作用： 日志打印工具，支持多种级别日志记录和格式化输出。

主要方法：

- `debug(...messages)`：记录调试日志。
- `trace(...messages)`：记录追踪级别的日志消息。
- `warn(...messages)`：记录警告日志。
- `error(...messages)`：记录错误日志。
- `info(...messages)`：记录信息日志。
- `log(...messages)`：记录默认日志。

详细文档： [ConsoleJS类](#)

CoreApi

作用： 核心实体管理接口，用于获取、注册、删除和管理产品下的设备实体，以及同步订阅资源和数字孪生操作。

主要方法：

- `getEntityAttribute(productId, deviceId)`：获取实体属性。
- `getEntityInfo(productId, deviceId)`：获取实体信息。
- `getConstituteEntityInfo(productId, deviceId)`：获取构成实体信息。
- `registerEntity(productId, entity)`：注册实体。
- `deleteEntity(productId, deviceIds)`：删除实体。
- `getEntityList(productId)`：获取实体列表。
- `getEntityCacheList(productId)`：获取实体缓存列表。
- `syncSubscriptionResource()`：同步订阅资源。
- `addEntityToTwin(productId, twinId, entity)`：将实体添加到数字孪生。

详细文档： [CoreApi类](#)

FileStorage

作用： 处理文件存储和缓存，支持 Base64 数据存储到磁盘或 Redis，返回访问链接。

主要方法：

- `disk(base64Encode, ...params)`：存储到磁盘。
- `cache(base64Encode, ...params)`：缓存到 Redis。

详细文档： [FileStorage类](#)

MockXMLHttpRequest

作用： 模拟 XMLHttpRequest，用于发起 HTTP 请求，支持设置头、发送数据和获取响应（当前同步模式）。

主要方法：

- `open(method, url)`：初始化请求。
- `open(method, url, async)`：初始化请求（异步标志当前忽略）。
- `setRequestHeader(header, value)`：设置单个请求头。
- `setRequestHeader(headers)`：批量设置请求头。
- `send()`：发送请求（无数据）。
- `send(data)`：发送请求（带数据）。
- `getReadyState()`：获取请求状态。
- `getResponseText()`：获取响应内容。
- `getStatus()`：获取状态码。
- `getStatusText()`：获取状态文本。

详细文档： [MockXMLHttpRequest类](#)

JtsGeometryUtils

作用： 封装 JTS 几何操作，用于创建、转换、计算几何对象，如点、线、多边形等。同时用于在图像上绘制多边形、标签和全局文本，支持自定义样式和颜色解析。

主要方法：

- `createCoordinates(coords)`：创建坐标数组，确保闭合。
- `normalizeCoordinates(coordinates, width, height)`：坐标归一化。
- `denormalizeCoordinates(coordinates, width, height)`：坐标去归一化。
- `createCoordinatesXYXY(coords)`：从 xyxy 格式创建坐标。
- `createCoordinatesXYWH(coords)`：从 xywh 格式创建坐标。
- `toCoordinatesByArray(coords)`：二维数组转坐标对象。
- `toCoordinates(coords)`：单个坐标转对象。

- `pixelToGeoCoordinates(...)` : 像素坐标转地理坐标。
- `pixelToGeoCoordinatesByScale(...)` : 使用比例尺转地理坐标。
- `createPoint(x, y)` : 创建点对象。
- `createLineString(coords)` : 创建线对象。
- `createPolygon(coordinates)` : 创建多边形对象。
- `isIntersects(geom1, geom2)` : 判断相交。
- `isContains(container, contained)` : 判断包含。
- `intersection(geom1, geom2)` : 计算交集。
- `intersectionArea(geom1, geom2)` : 计算交集面积。
- `union(geom1, geom2)` : 计算并集。
- `buffer(geometry, distance)` : 计算缓冲区。
- `distance(geom1, geom2)` : 计算距离。
- `isPointOnLine(lineString, point)` : 判断点在线上。
- `toText(geometry)` : 转 WKT 字符串。
- `drawBox(imageSource, boxes)` : 绘制多边形, 使用默认样式。
- `drawBox(imageSource, boxes, globalLabel, labels)` : 绘制多边形, 支持全局标签和单个标签。
- `drawBox(imageSource, boxes, lineColor, lineWidth, fillColor, globalLabel, labels, labelColor, labelBgColor, labelFillColor, labelFontSize)` : 绘制多边形, 支持完全自定义样式。
- `parseColor(colorStr)` : 解析颜色字符串。
- `parseColor(r, g, b)` : 从 RGB 值创建颜色对象。
- `parseColor(r, g, b, a)` : 从 RGBA 值创建颜色对象。

详细文档 : [JtsGeometryUtils类](#)

Console类

Console 接口文档

本文档提供了 `Console` 类的 JavaScript 接口。该类用于消息解析脚本日志打印，支持处理各种类型的参数，并将日志消息存储到列表中，供后续格式化输出。

类描述

`Console` 是一个日志打印工具，支持 `debug`、`warn`、`error`、`info` 和 `log` 五种日志级别，能够处理多种类型的输入参数（包括对象、错误对象和空值），并将日志消息存储到提供的列表中。日志消息支持 JSON 序列化，添加时间戳和日志级别。

方法

`debug(...messages)`

记录调试级别的日志消息。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入（对象、字符串、错误对象等）。

行为

- 将消息格式化为字符串并以 `DEBUG:` 前缀添加到 `loggedMessages` 数组。

示例

```
console.debug("Test message", { key: "value" }, null);  
// loggedMessages 包含: "DEBUG: \"Test message\" {\"key\":\"value\"} null"
```

`trace(...messages)`

记录追踪级别的日志消息（用于函数调用栈追踪或详细执行路径）。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入（对象、字符串、错误对象等）。

行为

- 将消息格式化为字符串并以 `TRACE:` 前缀添加到 `loggedMessages` 数组。
- 在后续格式化输出时，日志级别显示为 `TRACE`。

示例

```
consoleJS.trace("Entering function processData", { input: data });  
// loggedMessages 包含: "TRACE: \\"Entering function processData\\" {\\"input\\":...}"
```

warn(...messages)

记录警告级别的日志消息。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入。

行为

- 将消息格式化为字符串并以 `WARN:` 前缀添加到 `loggedMessages` 数组。

示例

```
console.warn("Warning occurred", { code: 404 });  
// loggedMessages 包含: "WARN: \\"Warning occurred\\" {\\"code\\":404}"
```

error(...messages)

记录错误级别的日志消息。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入。

行为

- 将消息格式化为字符串并以 `ERROR:` 前缀添加到 `loggedMessages` 数组。
- 如果参数是错误对象 (`Error`)，会包含错误消息和堆栈跟踪。

示例

```
console.error("Something went wrong");  
// loggedMessages 包含: "ERROR: Something went wrong\n\tat ..."
```

info(...messages)

记录信息级别的日志消息。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入。

行为

- 将消息格式化为字符串并以 `INFO:` 前缀添加到 `loggedMessages` 数组。

示例

```
console.info("Operation successful", { status: "OK" });  
// loggedMessages 包含: "INFO: \"Operation successful\" {\"status\":\"OK\"}"
```

`log(...messages)`

记录默认级别的日志消息。

参数

- `...messages` (`any[]`): 可变参数，支持任意类型的输入。

行为

- 将消息格式化为字符串（无前缀）添加到 `loggedMessages` 数组。

示例

```
console.log("General log", 123);  
// loggedMessages 包含: "\"General log\" 123"
```

注意事项

- 消息处理：
 - 如果参数为 `null`，记录为字符串 `"null"`。
 - 如果参数是错误对象（`Error`），记录错误消息和完整堆栈跟踪。
 - 其他类型的参数尝试使用 JSON 序列化，若失败则直接使用 `toString()`。
 - 所有消息以空格分隔并去除末尾空格。

CoreApi类

CoreApi JavaScript 接口文档

本文档提供了 `CoreApi` 类的 JavaScript 接口，基于提供的 Java 方法签名翻译而来。该接口主要用于管理产品下的设备实体（Entity），包括查询、注册、删除、列表获取、同步以及数字孪生相关的操作。所有返回 JSON 数据的方法在 JavaScript 中对应返回普通对象（Object）或对象数组。

类描述

`CoreApi` 是核心实体管理接口，用于物联网或设备管理平台中对产品（Product）和设备（Device/Entity）的生命周期和数据进行操作。

方法

`getEntityAttribute(productId, deviceId)`

获取指定设备实体的属性信息。

参数

- `productId` (string): 产品 ID。
- `deviceId` (string): 设备（实体）ID。

返回值

- `Object` : 包含该设备实体属性的 JSON 对象。

示例

```
const attributes = CoreApi.getEntityAttribute("prod_123", "dev_456");
// 返回: { temperature: 25.5, humidity: 60, ... }
```

`getEntityInfo(productId, deviceId)`

获取指定设备实体的基本信息。

参数

- `productId` (string): 产品 ID。
- `deviceId` (string): 设备（实体）ID。

返回值

- `Object` : 包含该设备实体基本信息的 JSON 对象（如名称、状态、在线状态等）。

示例

```
const info = CoreApi.getEntityInfo("prod_123", "dev_456");  
// 返回: { name: "传感器A", status: "online", ... }
```

`getConstituteEntityInfo(productId, deviceId)`

获取指定设备实体的构成实体信息（常用于组合设备或子设备场景）。

参数

- `productId` (`string`): 产品 ID。
- `deviceId` (`string`): 设备（实体）ID。

返回值

- `Object` : 包含构成实体信息的 JSON 对象。

示例

```
const constituteInfo = CoreApi.getConstituteEntityInfo("prod_123", "dev_456");  
// 返回: { components: [...], subDevices: [...] }
```

`registerEntity(productId, entitys)`

注册多个新设备实体到指定产品下。

参数

- `productId` (`string`): 产品 ID。
- `entitys` (`Array<Object>`): 实体信息列表，每个元素为一个 Map（对象），包含设备注册所需字段（如 `deviceId`、`name` 等）。

返回值

- `Object` : 注册结果的 JSON 对象（通常包含成功/失败信息）。

示例

```
const entitys = [  
  { deviceId: "dev_001", name: "设备1" },  
  { deviceId: "dev_002", name: "设备2" }  
];  
const result = CoreApi.registerEntity("prod_123", entitys);  
// 返回: { success: true, failed: [] }
```

deleteEntity(productId, deviceIds)

删除指定产品下的多个设备实体。

参数

- productId (string): 产品 ID。
- deviceIds (Array<string>): 要删除的设备 ID 列表。

返回值

- Object: 删除结果的 JSON 对象（通常包含成功/失败信息）。

示例

```
const deviceIds = ["dev_001", "dev_002"];
const result = CoreApi.deleteEntity("prod_123", deviceIds);
// 返回: { success: true, deletedCount: 2 }
```

getEntityList(productId)

获取指定产品下的所有设备实体列表。

参数

- productId (string): 产品 ID。

返回值

- Array<Object>: 设备实体信息对象数组。

示例

```
const list = CoreApi.getEntityList("prod_123");
// 返回: [{ deviceId: "dev_001", name: "设备1", ... }, ...]
```

getEntityCacheList(productId)

获取指定产品下的设备实体缓存列表（可能来自本地缓存或快速查询通道）。

参数

- productId (string): 产品 ID。

返回值

- Array<Object>: 缓存的设备实体信息对象数组。

示例

```
const cacheList = CoreApi.getEntityCacheList("prod_123");  
// 返回: [{ deviceId: "dev_001", ... }, ...]
```

syncSubscriptionResource()

同步订阅资源（通常用于推送订阅、物模型同步或资源刷新）。

参数

- 无参数

返回值

- `void`：无返回值。

示例

```
CoreApi.syncSubscriptionResource();  
// 执行同步操作
```

addEntityToTwin(productId, twinId, entitys)

将多个实体添加到指定的数字孪生（Twin）中。

参数

- `productId` (`string`): 产品 ID。
- `twinId` (`string`): 数字孪生 ID。
- `entitys` (`Array<Object>`): 要添加的实体信息列表，每个元素为一个 Map（对象）。

返回值

- `Object`：添加结果的 JSON 对象。

示例

```
const entitys = [  
  { deviceId: "dev_001", role: "sensor" },  
  { deviceId: "dev_002", role: "actuator" }  
];  
const result = CoreApi.addEntityToTwin("prod_123", "twin_001", entitys);  
// 返回: { success: true, addedCount: 2 }
```

注意事项

- 所有返回 `JSONObject` 的方法在 JavaScript 中对应返回普通对象（`Object`）。

- 返回 `List<JSONObject>` 的方法对应返回对象数组（ `Array<Object>` ）。
- 参数中的 `List<Map<String, Object>>` 在 JavaScript 中使用数组 + 对象表示。
- 建议在使用前检查参数有效性（如 productId、deviceId 不为空）。
- 操作结果对象通常包含成功标志、错误信息或统计数据，调用方应根据实际业务处理返回值。

FileStorage类

FileStorage 类文档

本文档提供了 `FileStorage` 类的 JavaScript 接口。该类用于处理文件的存储和缓存，支持将 Base64 编码的数据存储到磁盘或缓存，并返回文件的访问链接。

类描述

`FileStorage` 提供两种文件存储方式：磁盘存储（`disk` 方法）和 缓存存储（`cache` 方法）。磁盘存储将 Base64 编码的数据解码后写入文件系统，缓存存储将数据直接存储到缓存中。两种方式均支持设置文件后缀和名称，并通过异步线程执行存储操作以提高性能。

方法

`disk(base64Encode, ...params)`

将 Base64 编码的字符串解码并写入磁盘文件。

参数

- `base64Encode` (`string`): Base64 编码的字符串，包含文件内容。
- `...params` (`string[]`): 可变参数，最多接受两个参数：
- `params[0]` (`string`): 文件后缀（如 `jpg`、`png`）。
- `params[1]` (`string`, 可选): 文件名称。如果未提供，使用 8 位 UUID 作为文件名；如果提供，将覆盖同名文件。

返回值

- `string`: 文件的访问链接，基于网关访问 URL 和文件路径生成。

异常

- 如果消息解析模拟输入，则抛出错误：`"当前模拟输入暂不支持文件存储！"`。
- 如果 `base64Encode` 或 `params` 为空或无效，抛出错误：`"参数不能为空"`。
- 如果文件大小超过 100MB，抛出错误：`"文件大小超过限制，请上传小于 100MB 的文件"`。
- 如果解码或写入文件失败，抛出错误：`"将 Base64 编码的字符串解码并写入指定路径的文件时出错"`。

行为

- 异步解码 Base64 数据并写入文件，文件路径格式为：`{yyyyMMdd}/{fileName}.{suffix}`。
- 记录文件过期时间，默认为7天。
- 返回的访问链接。

示例

```
const base64Encode = "data:image/jpeg;base64,...";
const fileUrl = FileStorage.disk(base64Encode, "jpg");
// 返回: "http://gateway-url/uploadFile/20250416/xxxxxxx.jpg"

const fileUrlWithName = FileStorage.disk(base64Encode, "jpg", "customImage");
// 返回: "http://gateway-url/uploadFile/20250416/customImage.jpg"
```

cache(base64Encode, ...params)

将 Base64 编码的字符串存储到缓存中。

参数

- `base64Encode` (`string`): Base64 编码的字符串，包含文件内容。
- `...params` (`string[]`): 可变参数，最多接受两个参数：
- `params[0]` (`string`): 文件后缀（如 `jpg`、`png`）。
- `params[1]` (`string` , 可选): 文件名称。如果未提供，使用 8 位 UUID 作为文件名；如果提供，将覆盖同名文件。

返回值

- `string`: 文件的访问链接，格式为 `/api/cache/download/" + fileName`。

异常

- 如果 `base64Encode` 或 `params` 为空或无效，抛出错误：`"参数不能为空"`。
- 如果文件大小超过 10MB，抛出错误：`"文件大小超过限制，请上传小于 10MB 的文件或使用disk存储"`。
- 如果缓存失败，抛出错误：`"缓存资源数据时出错"`。

行为

- 异步解码 Base64 数据以检查大小，之后将原始 Base64 字符串存储到缓存。
- 记录文件过期时间，默认为7天。
- 返回的访问链接。

示例

```
const base64Encode = "data:image/jpeg;base64,...";
const fileUrl = FileStorage.cache(base64Encode, "jpg");
// 返回: "http://gateway-url/api/cache/download/xxxxxxx.jpg"

const fileUrlWithKey = FileStorage.cache(base64Encode, "jpg", "customKey");
// 返回: "http://gateway-url/api/cache/download/customKey.jpg"
```

注意事项

- 文件大小限制：
- `disk` 方法限制文件大小为 100MB。

- `cache` 方法限制文件大小为 10MB，超出时建议使用 `disk` 方法。

MockXMLHttpRequest类

MockXMLHttpRequest 类文档

本文档提供了 `MockXMLHttpRequest` 类的 JavaScript 接口。该类模拟了浏览器中 `XMLHttpRequest` 的功能，用于发起 HTTP 请求，支持设置请求方法、URL、请求头以及发送数据，并处理响应。

类描述

`MockXMLHttpRequest` 是一个模拟 HTTP 请求的工具类，支持 `GET`、`POST` 等请求方法，能够设置请求头、发送数据（字符串或对象），并获取响应状态、状态文本和响应内容。异步执行当前暂不支持，请求以同步方式处理。

方法

`open(method, url)`

初始化 HTTP 请求。

参数

- `method` (`string`): HTTP 请求方法（如 `"GET"`、`"POST"`）。
- `url` (`string`): 请求的目标 URL。

行为

- 设置请求方法和 URL。

示例

```
req.open("GET", "https://api.example.com/data");
```

`open(method, url, async)`

初始化 HTTP 请求，带异步标志（当前异步暂不支持）。

参数

- `method` (`string`): HTTP 请求方法。
- `url` (`string`): 请求的目标 URL。
- `async` (`boolean`): 是否异步执行（当前忽略，固定为同步）。

行为

- 设置请求方法、URL 和异步标志。

示例

```
req.open("POST", "https://api.example.com/submit", false);
```

setRequestHeader(header, value)

设置单个请求头。

参数

- **header** (`string`): 请求头名称 (如 `"Content-Type"`) 。
- **value** (`string`): 请求头值 (如 `"application/json"`) 。

行为

- 仅在 `req.open` 后调用有效，将请求头添加到内部存储。

示例

```
req.open("POST", "https://api.example.com/submit");  
req.setRequestHeader("Content-Type", "application/json");
```

setRequestHeader(headers)

批量设置请求头。

参数

- **headers** (`Object`): 键值对对象，键为请求头名称，值为请求头值。

行为

- 仅在 `req.open` 后调用有效，将所有请求头添加到内部存储。

示例

```
req.open("POST", "https://api.example.com/submit");  
req.setRequestHeader({  
  "Content-Type": "application/json",  
  "Authorization": "Bearer token123"  
});
```

send()

发送 HTTP 请求（无请求体）。

行为

- 如果未调用 req.open ，抛出错误： "Invalid state: the request has not been opened." 。
- 使用设置的 method 、URL 和请求头发起 HTTP 请求。
- 更新 status （HTTP 状态码）、statusText （状态文本）、responseText （响应内容），并将 readyState 设置为 DONE （4）。
- 如果请求失败，抛出错误： "Error sending request: ..." 。

示例

```
req.open("GET", "https://api.example.com/data");
req.send();
console.log(req.responseText); // 输出响应内容
```

send(data)

发送 HTTP 请求（带请求体）。

参数

- data (any): 请求体数据，可以是字符串或对象。
- 如果是字符串，直接发送。
- 如果是对象，使用 JSON 序列化后发送。

行为

- 如果未调用 req.open ，抛出错误： "Invalid state: the request has not been opened." 。
- 设置 Content-Type （如果未设置）并发送请求体。
- 更新 status 、statusText 、responseText ，并将 readyState 设置为 DONE （4）。
- 如果请求失败，抛出错误： "Error sending request: ..." 。

示例

```
req.open("POST", "https://api.example.com/submit");
req.setRequestHeader("Content-Type", "application/json");
req.send({ key: "value" });
console.log(req.responseText); // 输出响应内容
```

getReadyState()

获取当前请求状态。

返回值

- number : 当前的 readyState 值（0 - 4）。

示例

```
req.open("GET", "https://api.example.com/data");  
console.log(req.readyState); // 输出: 1
```

getResponseText()

获取响应内容。

返回值

- `string | null` : 服务器返回的响应内容，请求未完成时为 `null`。

示例

```
req.open("GET", "https://api.example.com/data");  
req.send();  
console.log(req.responseText()); // 输出响应内容
```

getStatus()

获取 HTTP 状态码。

返回值

- `number` : HTTP 状态码（如 `200`、`404`），初始为 `0`。

示例

```
req.open("GET", "https://api.example.com/data");  
req.send();  
console.log(req.getStatus()); // 输出: 200（成功时）
```

getStatusText()

获取 HTTP 状态文本。

返回值

- `string | null` : HTTP 状态文本（如 `"OK"`、`"Not Found"`），初始为 `null`。

示例

```
req.open("GET", "https://api.example.com/data");  
req.send();  
console.log(req.getStatusText()); // 输出: "OK"（成功时）
```

注意事项

- 异步支持**：当前版本不支持异步请求（`async` 参数被忽略，请求以同步方式执行）。

- **请求头设置**：必须在调用 `open` 后、`send` 前设置请求头，否则无效。
- **数据处理**：非字符串类型的 `send(data)` 参数会自动 JSON 序列化。
- **异常处理**：
 - 如果请求未打开，`send` 方法抛出 "Invalid state: the request has not been opened."。
 - 如果请求失败（网络错误、URL 无效等），抛出 "Error sending request: ..."。
- **响应处理**：响应内容以字符串形式存储在 `responseText` 中，需自行解析（如 JSON 格式）。

JtsGeometry类

JtsGeometry 接口文档

本文档提供了 `JtsGeometry` 类的 JavaScript 接口。该类封装了 JTS 库的常见几何操作方法，用于处理点、线、多边形等几何对象的创建、转换、归一化、地理坐标转换以及几何计算等功能。同时该接口描述了在图像上绘制多边形、标签和全局文本的方法，支持基于 URL 和 Base64 编码的图像。

方法

`createCoordinates(coords)`

根据坐标列表创建坐标数组。

参数

- `coords` (`Array<Array<number>>`): 坐标列表，每个元素是一个长度为 2 或 3 的数组，分别表示 x 和 y 坐标（可选 z 坐标）。

返回值

- `Array<{x: number, y: number, z?: number}>`: 坐标对象数组，确保多边形闭合（起点和终点相同）。

注意事项

- 如果输入坐标未闭合（起点和终点不同），会自动添加起点坐标作为终点以闭合多边形。

示例

```
const coords = [[100, 100], [200, 100], [200, 200]];
const coordinates = JtsGeometry.createCoordinates(coords);
// 返回: [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 100}]
```

`normalizeCoordinates(coordinates, width, height)`

对坐标数组进行归一化处理。

参数

- `coordinates` (`Array<{x: number, y: number}>`): 要归一化的坐标对象数组。
- `width` (`number`): 归一化参考宽度。
- `height` (`number`): 归一化参考高度。

返回值

- `Array<{x: number, y: number}>` : 归一化后的坐标对象数组，x 和 y 坐标分别除以宽度和高度。

示例

```
const coordinates = [{x: 100, y: 100}, {x: 200, y: 200}];
const normalized = JtsGeometry.normalizeCoordinates(coordinates, 400, 300);
// 返回: [{x: 0.25, y: 0.333...}, {x: 0.5, y: 0.666...}]
```

denormalizeCoordinates(coordinates, width, height)

对归一化的坐标数组进行去归一化处理。

参数

- `coordinates` (`Array<{x: number, y: number}>`): 归一化的坐标对象数组。
- `width` (`number`): 去归一化参考宽度。
- `height` (`number`): 去归一化参考高度。

返回值

- `Array<{x: number, y: number}>` : 去归一化后的坐标对象数组，x 和 y 坐标分别乘以宽度和高度。

示例

```
const coordinates = [{x: 0.25, y: 0.333}, {x: 0.5, y: 0.666}];
const denormalized = JtsGeometry.denormalizeCoordinates(coordinates, 400, 300);
// 返回: [{x: 100, y: 99.9}, {x: 200, y: 199.8}]
```

createCoordinatesXYXY(coords)

将 xyxy 格式的坐标转换为多边形坐标数组。

参数

- `coords` (`Array<number>`): 长度为 4 的数组，分别表示左上角 x、左上角 y、右下角 x、右下角 y。

返回值

- `Array<{x: number, y: number}>` : 包含 5 个坐标对象的数组（左上、右上、右下、左下、左上），表示闭合多边形。

示例

```
const coords = [100, 100, 200, 200];
const coordinates = JtsGeometry.createCoordinatesXYXY(coords);
// 返回: [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}, {x: 100, y: 100}]
```

createCoordinatesXYWH(coords)

将 xywh 格式的坐标转换为多边形坐标数组。

参数

- `coords` (`Array<number>`): 长度为 4 的数组，分别表示中心点 x、中心点 y、宽度、高度。

返回值

- `Array<{x: number, y: number}>`: 包含 5 个坐标对象的数组（左上、右上、右下、左下、左上），表示闭合多边形。

示例

```
const coords = [150, 150, 100, 100];
const coordinates = JtsGeometry.createCoordinatesXYWH(coords);
// 返回: [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}, {x: 100, y: 100}]
```

toCoordinatesByArray(coords)

将二维数组转换为坐标对象数组。

参数

- `coords` (`Array<Array<number>>`): 坐标列表，每个元素是长度为 2 或 3 的数组，分别表示 x 和 y 坐标（可选 z 坐标）。

返回值

- `Array<{x: number, y: number, z?: number}>`: 坐标对象数组。

示例

```
const coords = [[100, 100], [200, 200, 10]];
const coordinates = JtsGeometry.toCoordinatesByArray(coords);
// 返回: [{x: 100, y: 100}, {x: 200, y: 200, z: 10}]
```

toCoordinates(coords)

将单个坐标数组转换为坐标对象。

参数

- `coords` (`Array<number>`): 长度为 2 或 3 的数组，分别表示 x 和 y 坐标（可选 z 坐标）。

返回值

- `{x: number, y: number, z?: number}`: 坐标对象。

二、

示例

```
const coords = [100, 100, 10];
const coordinate = JtsGeometry.toCoordinates(coords);
// 返回: {x: 100, y: 100, z: 10}
```

pixelToGeoCoordinates(pixelCoordinates, pixelOrigin, geoOrigin, geoDistancePerPixelX, geoDistancePerPixelY)

将像素坐标转换为地理坐标。

参数

- pixelCoordinates** (`Array<{x: number, y: number}>`): 像素坐标对象数组。
- pixelOrigin** (`{x: number, y: number}`): 像素原点坐标。
- geoOrigin** (`{x: number, y: number, z?: number}`): 地理原点坐标。
- geoDistancePerPixelX** (`number`): 每个像素在 x 方向的地理距离。
- geoDistancePerPixelY** (`number`): 每个像素在 y 方向的地理距离。

返回值

- `Array<{x: number, y: number, z?: number}>` : 地理坐标对象数组。

示例

```
const pixelCoordinates = [{x: 100, y: 100}];
const pixelOrigin = {x: 0, y: 0};
const geoOrigin = {x: 10, y: 20};
const geoCoordinates = JtsGeometry.pixelToGeoCoordinates(pixelCoordinates, pixelOrigin, geoOrigin, 0.1, 0.2);
// 返回: [{x: 20, y: 40}]
```

pixelToGeoCoordinatesByScale(pixelCoordinates, pixelOrigin, geoOrigin, scale)

使用统一比例尺将像素坐标转换为地理坐标。

参数

- pixelCoordinates** (`Array<{x: number, y: number}>`): 像素坐标对象数组。
- pixelOrigin** (`{x: number, y: number}`): 像素原点坐标。
- geoOrigin** (`{x: number, y: number, z?: number}`): 地理原点坐标。
- scale** (`number`): 统一比例尺。

返回值

- `Array<{x: number, y: number, z?: number}>` : 地理坐标对象数组。

示例

```
const pixelCoordinates = [{x: 100, y: 100}];
const pixelOrigin = {x: 0, y: 0};
const geoOrigin = {x: 10, y: 20};
```

```
const geoCoordinates = JtsGeometry.pixelToGeoCoordinatesByScale(pixelCoordinates, pixelOrigin, geoOrigin, 0.1);  
// 返回: [{x: 20, y: 30}]
```

createPoint(x, y)

创建点对象。

参数

- **x** (**number**): 点的 x 坐标。
- **y** (**number**): 点的 y 坐标。

返回值

- **Point** : 点对象。

示例

```
const point = JtsGeometry.createPoint(100, 100);  
// 返回: Point 对象
```

createLineString(coords)

创建线对象。

参数

- **coords** (**Array<Array<number>>**): 坐标列表，每个元素是长度为 2 的数组，分别表示 x 和 y 坐标。

返回值

- **LineString** : 线对象。

示例

```
const coords = [[100, 100], [200, 200]];
const line = JtsGeometry.createLineString(coords);
// 返回: LineString 对象
```

createPolygon(coordinates)

创建多边形对象。

参数

- **coordinates** (**Array<{x: number, y: number}>**): 多边形的坐标对象数组，需闭合（起点和终点相同）。

返回值

- `Polygon` : 多边形对象。

示例

```
const coordinates = [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}, {x: 100, y: 100}];
const polygon = JtsGeometry.createPolygon(coordinates);
// 返回: Polygon 对象
```

`isIntersects(geom1, geom2)`

判断两个几何对象是否相交。

参数

- `geom1` (`Geometry`): 第一个几何对象。
- `geom2` (`Geometry`): 第二个几何对象。

返回值

- `boolean` : 如果相交返回 `true` , 否则返回 `false` 。

示例

```
const geom1 = JtsGeometry.createPolygon([...]);
const geom2 = JtsGeometry.createPolygon([...]);
const intersects = JtsGeometry.isIntersects(geom1, geom2);
// 返回: true 或 false
```

`isContains(container, contained)`

判断一个几何对象是否包含另一个几何对象。

参数

- `container` (`Geometry`): 包含者几何对象。
- `contained` (`Geometry`): 被包含者几何对象。

返回值

- `boolean` : 如果包含返回 `true` , 否则返回 `false` 。

示例

```
const container = JtsGeometry.createPolygon([...]);
const contained = JtsGeometry.createPoint(150, 150);
const contains = JtsGeometry.isContains(container, contained);
// 返回: true 或 false
```

intersection(geom1, geom2)

计算两个几何对象的交集。

参数

- `geom1` (`Geometry`): 第一个几何对象。
- `geom2` (`Geometry`): 第二个几何对象。

返回值

- `Geometry`: 交集的几何对象。

示例

```
const geom1 = JtsGeometry.createPolygon([...]);
const geom2 = JtsGeometry.createPolygon([...]);
const intersection = JtsGeometry.intersection(geom1, geom2);
// 返回: Geometry 对象
```

intersectionArea(geom1, geom2)

计算两个几何对象的交集面积。

参数

- `geom1` (`Geometry`): 第一个几何对象。
- `geom2` (`Geometry`): 第二个几何对象。

返回值

- `number`: 交集的面积。

示例

```
const geom1 = JtsGeometry.createPolygon([...]);
const geom2 = JtsGeometry.createPolygon([...]);
const area = JtsGeometry.intersectionArea(geom1, geom2);
// 返回: 面积值
```

union(geom1, geom2)

计算两个几何对象的并集。

参数

- `geom1` (`Geometry`): 第一个几何对象。
- `geom2` (`Geometry`): 第二个几何对象。

返回值

- `Geometry` : 并集的几何对象。

示例

```
const geom1 = JtsGeometry.createPolygon([...]);
const geom2 = JtsGeometry.createPolygon([...]);
const union = JtsGeometry.union(geom1, geom2);
// 返回: Geometry 对象
```

`buffer(geometry, distance)`

计算几何对象的缓冲区。

参数

- `geometry` (`Geometry`): 几何对象。
- `distance` (`number`): 缓冲区距离。

返回值

- `Geometry` : 缓冲区的几何对象。

示例

```
const geometry = JtsGeometry.createPoint(100, 100);
const buffer = JtsGeometry.buffer(geometry, 10);
// 返回: Geometry 对象
```

`distance(geom1, geom2)`

计算两个几何对象之间的距离。

参数

- `geom1` (`Geometry`): 第一个几何对象。
- `geom2` (`Geometry`): 第二个几何对象。

返回值

- `number` : 两个几何对象之间的距离。

示例

```
const geom1 = JtsGeometry.createPoint(100, 100);
const geom2 = JtsGeometry.createPoint(200, 200);
const distance = JtsGeometry.distance(geom1, geom2);
// 返回: 距离值
```

isPointOnLine(lineString, point)

判断点是否在线上。

参数

- `lineString` (`LineString`): 线对象。
- `point` (`Point`): 点对象。

返回值

- `boolean` : 如果点在线上返回 `true` , 否则返回 `false` 。

示例

```
const line = JtsGeometry.createLineString([[100, 100], [200, 200]]);
const point = JtsGeometry.createPoint(150, 150);
const isOnLine = JtsGeometry.isPointOnLine(line, point);
// 返回: true 或 false
```

toText(geometry)

将几何对象转换为 WKT 字符串。

参数

- `geometry` (`Geometry`): 几何对象。

返回值

- `string` : WKT 格式的字符串。

示例

```
const geometry = JtsGeometry.createPoint(100, 100);
const wkt = JtsGeometry.toText(geometry);
// 返回: "POINT (100 100)"
```

drawBox(imageSource, boxes)

在图像上绘制多边形，使用默认样式。

参数

- `imageSource` (`string`): 图像来源，可以是 URL (以 `http://` 或 `https://` 开头) 或 Base64 编码字符串 (包含或不包含 `data:image/...;base64,` 前缀)。
- `boxes` (`Array<Array<{x: number, y: number}>>`): 多边形数组，每个多边形是一个包含 `x` 和 `y` 属性的坐标对象数组。

返回值

- `string | null` : 结果图像的 Base64 编码字符串（不包含 `data:image/jpeg;base64,` 前缀），如果图像无法处理则返回 `null`。

默认样式

- 边框颜色: RGB(255, 245, 51)
- 边框宽度: 2 像素
- 填充颜色: RGBA(255, 245, 158, 0.5)
- 标签文字颜色: 黑色
- 标签背景颜色: RGB(255, 177, 177)
- 标签填充颜色: RGB(255, 216, 216)
- 标签字体大小: 12px

示例

```
const imageSource = "data:image/jpeg;base64,...";
const boxes = [
  [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}]
];
const result = JtsGeometry.drawBox(imageSource, boxes);
```

`drawBox(imageSource, boxes, globalLabel, labels)`

在图像上绘制多边形，可选择添加全局标签和每个多边形的标签，使用默认样式。

参数

- `imageSource` (`string`) : 图像来源，可以是 URL 或 Base64 编码字符串。
- `boxes` (`Array<Array<{x: number, y: number}>>`) : 多边形数组，每个多边形是一个坐标对象数组。
- `globalLabel` (`string | null`) : 显示在图像左上角的全局标签。如果为 `null` 或空，则不绘制全局标签。
- `labels` (`Array<string> | null`) : 每个多边形对应的标签数组。如果为 `null` 或长度小于 `boxes`，缺失的标签视为 `null`。

返回值

- `string | null` : 结果图像的 Base64 编码字符串，如果图像无法处理则返回 `null`。

默认样式

- 与 `drawBox(imageSource, boxes)` 方法相同。

示例

```
const imageSource = "https://example.com/image.jpg";
const boxes = [
```

```
const boxes = [
  [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}]
];
const globalLabel = "全局标签";
const labels = ["多边形 1"];
const result = JtsGeometry.drawBox(imageSource, boxes, globalLabel, labels);
```

drawBox(imageSource, boxes, lineColor, lineWidth, fillColor, globalLabel, labels, labelColor, labelBgColor, labelFillColor, labelFontSize)

在图像上绘制多边形，支持完全自定义的样式、标签和全局文本。

参数

- imageSource** (string)：图像来源，可以是 URL 或 Base64 编码字符串。
- boxes** (Array<Array<{x: number, y: number}>>)：多边形数组，每个多边形是一个坐标对象数组。
- lineColor** (string | Object)：多边形边框颜色。可以是十六进制字符串（例如 "#FFFF33"）或 RGB/A 对象（例如 {r: 255, g: 245, b: 51} 或 {r: 255, g: 245, b: 51, a: 1.0}）。
- lineWidth** (number)：多边形边框宽度（像素）。
- fillColor** (string | Object)：多边形填充颜色。可以是十六进制字符串或 RGB/A 对象。
- globalLabel** (string | null)：显示在图像左上角的全局标签。如果为 null 或空，则不绘制全局标签。
- labels** (Array<string> | null)：每个多边形对应的标签数组。如果为 null 或长度小于 boxes，缺失的标签视为 null。
- labelColor** (string | Object)：标签文字颜色。可以是十六进制字符串或 RGB/A 对象。
- labelBgColor** (string | Object)：标签背景颜色。可以是十六进制字符串或 RGB/A 对象。
- labelFillColor** (string | Object)：标签背景边框颜色。可以是十六进制字符串或 RGB/A 对象。
- labelFontSize** (number)：标签字体大小（像素）。

返回值

- string | null：结果图像的 Base64 编码字符串，如果图像无法处理则返回 null。

注意事项

- 多边形绘制包含填充内部和边框。每个顶点用白色填充圆圈（半径 4px）标记，圆圈边框与 lineColor 一致。
- 标签在每个多边形的第一个坐标处绘制，带有圆角矩形背景（圆角半径 8px）和内边距（水平 8px，垂直 2px）。
- 如果第一个坐标靠近图像边缘（10px 以内），标签会调整以保持图像范围内。
- 全局标签在图像左上角绘制，带有圆角矩形背景，字体大小为 labelFontSize 的四倍。
- 启用抗锯齿以改善多边形和圆圈的渲染平滑度。
- 使用的字体为 "Microsoft YaHei"，加粗样式。

示例

```
const imageSource = "data:image/jpeg;base64,...";
const boxes = [
  [{x: 100, y: 100}, {x: 200, y: 100}, {x: 200, y: 200}, {x: 100, y: 200}]
];
const lineColor = "#FFFF33";
const lineWidth = 3;
const fillColor = {r: 255, g: 245, b: 158, a: 0.5};
const globalLabel = "全局标签";
const labels = ["多边形 1"];
const labelColor = "#000000";
```

```
const labelColor = #000000 ;
const labelBgColor = "#FFB1B1";
const labelFillColor = "#FFD8D8";
const labelFontSize = 14;
const result = JtsGeometry.drawBox(
  imageSource,
  boxes,
  lineColor,
  lineWidth,
  fillColor,
  globalLabel,
  labels,
  labelColor,
  labelBgColor,
  labelFillColor,
  labelFontSize
);
```

parseColor(colorStr)

将颜色字符串解析为颜色对象。

参数

- `colorStr` (`string`)：十六进制格式的颜色字符串（例如 `"#FF0000"` 或 `"0xFF0000"` ）。

返回值

- `Object | null`：包含 `r`、`g`、`b`（可选 `a`）属性的对象，如果颜色字符串无效则返回 `null`。

示例

```
const color = JtsGeometry.parseColor("#FF0000");
// 返回: {r: 255, g: 0, b: 0}
```

parseColor(r, g, b)

从 RGB 值创建颜色对象。

参数

- `r` (`number`)：红色分量 (0-255)。
- `g` (`number`)：绿色分量 (0-255)。
- `b` (`number`)：蓝色分量 (0-255)。

返回值

- `Object`：包含 `r`、`g`、`b` 属性的对象。

示例

```
const color = JtsGeometry.parseColor(255, 0, 0);
// 返回: {r: 255, g: 0, b: 0}
```

parseColor(r, g, b, a)

从 RGBA 值创建颜色对象。

参数

- **r** (**number**)：红色分量 (0-255)。
- **g** (**number**)：绿色分量 (0-255)。
- **b** (**number**)：蓝色分量 (0-255)。
- **a** (**number**)：透明度分量 (0.0-1.0)。

返回值

- **Object**：包含 **r**、**g**、**b** 和 **a** 属性的对象。

示例

```
const color = JtsGeometry.parseColor(255, 245, 158, 0.5);  
// 返回: {r: 255, g: 245, b: 158, a: 0.5}
```