Things 二开手册

版本号：V 2.0

2025 年 10 月 17 日

合沃物联技术(南京)有限公司

版本修订履历

修订日期	版本	修订者	修订内容
2025/10/16	V1.0	代银鹏	初稿制定
2025/10/17	V2.0	代银鹏	规整 OpenApi 接口、补充数据库开放内容
2025/10/17	V3.0	代银鹏	离线文档改为 pdf 格式
2025/12/20	V4.0	何海波	补充 3.6 章节 用户系统对接；4.6 章节 插件融合；第五章节 单点登录

目录

第一章 二开介绍

1.1 阅读对象

本文档用于指导您如何基于合沃物联出品的物联网开发平台 Things 进行二次开发使用说明。因此本文档面向的是：软件开发工程师、系统架构师及项目技术人员或其他软件开发领域的专业技术人员。读者需具备基本的编程知识并至少掌握任意一门编程语言。

1.2 二开介绍

二次开发是 Things 平台的核心能力之一。Things 平台通过提供高度灵活的插件系统、开放稳定的引擎服务、基础应用的 OpenApi、数据订阅等方式满足您在实际业务中所需的二次开发基础功能。通过插件机制，用户可以将自研应用、第三方服务或定制化功能无缝集成到 Things 平台中，实现业务逻辑的灵活扩展与系统能力的增强。

1.2 术语定义

术语	解释
插件	指基于合沃软件平台提供的开放能力，并按照合沃“插件”系统规范进行功能扩展而开发的应用
引擎	为应用提供支撑且能够独立部署运行的核心基础部分
单点登录	是一种身份认证方案，允许用户使用同一组凭证（如用户名和密码）登录多个相互信任的应用程序系统。用户只需登录一次，即可访问所有授权的应用，而无需在访问每个应用时重新登录
OpenAPI	是一个与编程语言无关的、机器可读的标准化格式，用于描述、创建、可视化和调用 RESTful API。它通过一个 YAML 或 JSON 文件来明确定义 API 的所有细节。
关系数据库	一种基于关系模型存储数据的数据库，数据被组织在由行和列构成的表中，并通过表之间的关系进行管理。它使用 SQL 语言进行数据操作和查询。
时序数据库	一种专门为处理时间序列数据而优化的数据库。时间序列数据是按时间顺序记录的一系列数据点，通常用于存储和高效查询带时间戳的监控指标、传感器读数等。

1.3 免责声明

44544

1.4 注意事项

454545

第二章 数据订阅

2.1、应用场景

数据订阅功能为第三方系统或插件业务模块提供了实时、可靠的数据接入能力，适用于多种业务集成与数据拉通场景。通过订阅平台推送的各类主题消息，用户可实现设备状态、实时数据、控制结果等信息的实时获取与处理，支撑上层业务系统的灵活构建与快速响应。

典型应用场景包括：

工厂系统（MES/ERP）：实时接收产线设备状态与工艺数据，驱动生产排程、物料调度等业务流。

能源管理系统：订阅能耗相关数据，用于能效分析、负荷预测与节能优化。

设备设施运维平台：实时获取设备告警、状态变化与控制反馈，实现预测性维护与远程运维。

2.2、使用说明

进入系统配置模块，在平台主菜单中选择【系统配置】入口，点击进入系统管理功能区域，访问凭证管理功能,在系统配置菜单中定位【凭证管理】选项，点击【添加凭证】或【新建凭证】按钮，系统弹出凭证创建表单对话框，下拉框选择数据订阅，设置认证用户名和密码。然后通过设置的用户名和密码进行消息订阅。

(1)用户名设置：在用户名输入框中指定唯一的认证标识

示例：输入 admin 作为 MQTT 连接用户名

要求：用户名需全局唯一，支持字母、数字、下划线组合

(2)密码设置：在密码输入框中设置连接认证密码

示例：设置 123456 作为 MQTT 连接密码

要求：密码需满足复杂度要求，长度 6-20 位字符

(3)生成并保存凭证

系统基于输入的用户名和密码生成唯一的数据订阅凭证

【连接示例】

mqtt 连接端口：1889

连接用户名账号：admin

连接账号密码：123456

连接 ip：iot.hiwooiot.cn

连接标识：ThirdParty_xxx

然后根据下面主题进行平台消息订阅

2.2.1 主题定义

主题分类	主题名	主题	QoS	是否保留	说明
设备信息订阅	设备位置	/${boxId}/real/location	1	False	
	网关信息	/box/config/${tenantId}/${boxId}	1	False	
	遗嘱消息	/${boxId}/willmessage	2	True	
设备数据订阅	实时数据推送	/clean/${boxId}/realtime	0	False	
控制结果订阅	原始控制数据	/commService/boxDatas/resp/${boxId}	2	False	

2.2.2 消息定义

2.2.2.1 设备信息订阅

（1）设备位置

描述	设备位置（4G、5G 网关支持）
Topic	/{boxId}/real/location
QoS	1

Payload	{
	"notiyLocation": {
	"longitude": 117.151245,
	"latitude": 30.532293
	}
	}

备注	notiyLocation：设备位置 longitude：经度 latitude：纬度

应用场景：

(1) 实时位置追踪: 用于监控设备的实时位置,比如实时船舶追踪

(2) 地理围栏报警: 当设备进出特定区域时触发警报

(3) 路径规划优化: 基于设备位置进行智能调度和路线规划

（2）网关信息

描述	网关配置信息
Topic	/box/config/${tenantId}/${boxId}
QoS	1
Payload	{ "profile": { "tenantId": "HW01010012", "userId": "1231" }, "basicInfo": { "boxId": "模板编号", "boxName": "设备名称", "source": "BoxPlugin", "operationType": "update/delete", "subDevices": [{ "deviceId": 1, "deviceName": "TCP", "portName": "lan", "variableData": [{ "type": "collect", "deviceId": 2, "dataId": "变量标识", "name": "变量名称", "optionType": "rw,读写：rw、只读：r、只写：w", "dataType": "bool，支持 bool、int、float、double、string", "decimal": "2", "min": "0", "max": "1000000", "unit": "帕/Pa，单位" }] }] }, , "events": [{

	"eventId": "报警唯一标识符", "name": "报警名称", "type": "warn：报警、 info：事件 ", "level": "一般警告,告警/事件级别用户可自定义，显示用户定义的字符", "triggerValue": "报警数值，数据推送时使用", "message": "报警/事件信息，数据推送时使用", "mode": "gateway:网关、device:设备、site:站点" }] }
备注	subDevices 和 events 可以分开推送，变量多于 5000 条会分批推送，如果同一设备多于 5000 条可能推送多次（多个设备列表，每个设备最多 5000 点）

应用场景：

(1) 配置审计与追踪：记录所有下发给网关的配置变更（创建、更新、删除），满足合规性要求。

(2) 配置同步与备份：将接收到的配置信息保存到备份数据库，防止配置丢失。

(3) 配置分析优化：分析所有网关的配置模式，为优化设备模板和标准化提供数据支持。

(4) 权限与配额检查：验证下发的配置是否符合租户的权限和变量数量配额限制。

(5) 配置一致性验证：将接收到的配置与系统中预期的配置进行比对，发现异常时告警。

(6) 驱动下游系统更新：当配置变更时，通知数据分析、报警等其他系统同步更新其设备模型。

（3）网关离线

描述	网关离线推送
Topic	/${boxId}/willmessage
QoS	1

Payload	{
	"BoxId":"3500224030414580051",
	"Status":"offline"
	}

备注	offline：网关离线

应用场景：

① 实时状态监控： 在监控大屏或地图上，立即将对应设备状态变更为“离线”。

② 触发告警通知： 向运维人员发送短信或 App 推送，告知设备异常下线。

③ 标记数据中断： 通知数据分析系统，此网关数据流已中断，需处理数据缺失。

④ 停用关联服务： 将该网关及其下属的所有设备标记为“不可用”，避免误操作。

2.2.2.2 设备数据订阅

描述	实时数据订阅
Topic	/clean/${boxId}/realtime
QoS	0
Payload	{ "identifier": "1962403053254430720", "properties": [{ "dataId": "1962403053342511104", "dataType": "2", "ts": "1759135076345", "value": "1" }, { "dataId": "1962403053342511105", "dataType": "2", "ts": "1759135076345", "value": "1" }, { "dataId": "1962403053342511106", "dataType": "2", "ts": "1759135076345", "value": "1" }, { "dataId": "1962403053342511107", "dataType": "2", "ts": "1759135076345", "value": "0" }], "tenantId": "HW6000001176646749", "type": "device" }
备注	type：设备类型 tenantId：租户 ID

	dataId：变量 ID value：变量值 ts：变量采集时间，时间戳形式，精确到毫秒 dataType：数据类型（1-bool、2-int、3-float、4-string、5-double）

应用场景：

(1) 实时监控大屏：将设备运行状态、工艺参数实时展示在指挥中心大屏上，提供全局态势感知。

(2) 实时业务逻辑：基于数据值触发自动化控制，如当温度超过阈值时自动开启冷却系统。

(3) 实时数据分析：对接流处理引擎，对设备数据进行即时分析和异常检测。

(4) 实时报警判断：结合配置的报警规则，对实时数据进行判断，触发即时告警。

(5) 运行状态看板：为车间管理人员提供产线实时运行状态的可视化看板。

(6) 质量控制监控：实时监控产品质量相关参数，及时发现生产偏差。

(7) 能效实时监测：监控设备能耗数据，为节能优化提供实时依据。

2.2.2.3 控制结果订阅

描述	反向控制结果
Topic	/commService/boxDatas/resp/${boxId}
QoS	2

Payload	{
	"result": 0,
	"param": {
	"type": "writeData"
	}
	}

备注	type：响应类型 result: 响应结果，0 成功 1 失败

应用场景：

(1) 控制指令确认：确认远程控制指令（如启停设备、修改参数）是否成功执行。

(2) 操作状态同步：将控制结果同步到监控界面，实时显示设备的最新状态。

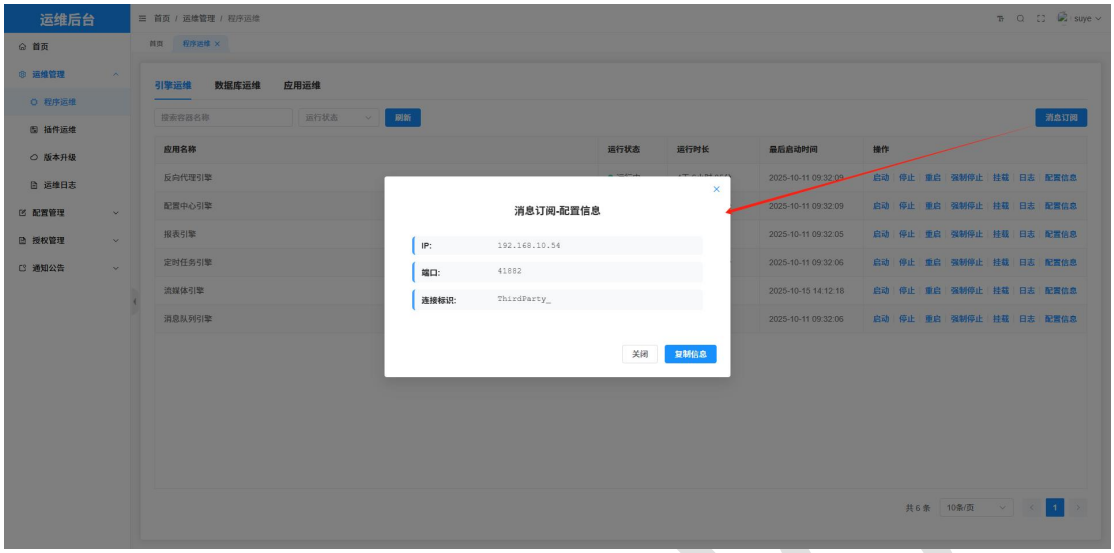
(3) 操作日志记录：记录所有控制操作的结果（成功/失败），用于审计和追溯。

(4) 失败告警触发：当控制失败（result≠0）时，立即通知运维人员介入处理。

(5) 业务流程推进：控制成功后触发下一阶段操作，如"设备启动成功→开始生产"。

(6) 用户操作反馈：在操作界面上立即向用户显示"操作成功"或"操作失败"的提示。

2.2.3 消息订阅连接信息配置获取



点击消息订阅获取消息订阅连接信息

2.2.4 使用示例

(1)配置步骤：

第一步：创建数据订阅凭证

在平台中创建专用凭证：

用户名：location_subscriber

密码：location@123456

第二步：第三方 MQTT 工具配置,以 MQTT 为例：

连接配置

Profile Name: 设备位置订阅

Broker Address: iot.hiwooiot.cn

Broker Port: 1889

Client ID: ThirdParty_Location_001

User Name: location_subscriber

Password: location@123456

2.2.4.1 获取网关设备位置示例

1、**场景描述**：需要获取网关设备（ID：84135354354355313132）的实时位置信息

（1）订阅主题：/84135354354355313132/real/location

QoS：1

（2）订阅消息接收：

```
{
  "notiyLocation": {
    "longitude": 117.151245,
    "latitude": 30.532293
  }
}
```

收到消息后，可以在第三方工具中看到：

经度：117.151245

纬度：30.532293

基于接收的位置数据，我们可以用来：

实时位置追踪，在地图系统中显示设备实时位置，更新设备轨迹记录

地理围栏报警，路径规划优化

2.2.4.2 网关信息订阅示例

1、场景描述：需要监控租户 HW01010012 下所有网关的配置信息

（1）订阅主题：/box/config/HW01010012/+

QoS：1

（2）预期接收消息示例

```
{
  "profile": {
    "tenantId": "HW01010012",
    "userId": "1231"
  },
  "basicInfo": {
    "boxId": "84135354354355313132",
    "boxName": "智能网关设备",
    "source": "BoxPlugin",
    "operationType": "update",
    "subDevices": [
      {
        "deviceId": 1,
```

```
            "deviceName": "温度传感器",
            "portName": "lan",
            "variableData": [{
              "type": "collect",
              "deviceId": 2,
              "dataId": "temp_001",
              "name": "环境温度",
              "optionType": "r",
              "dataType": "float",
              "decimal": "1",
              "min": "-20",
              "max": "60",
              "unit": "℃"
            }]
          }
        ]
    },
    "events": [
      {
        "eventId": "high_temp_alert",
        "name": "高温报警",
        "type": "warn",
        "level": "紧急",
        "triggerValue": "45",
        "message": "环境温度超过安全阈值",
        "mode": "device"
      }
    ]
}
```

2.2.4.3 网关离线监控示例

1、场景描述：实时监控网关（84135354354355313132）离线状态，及时触发告警

（1）订阅主题：/84135354354355313132/willmessage (QoS: 2)

（2）预期接收消息示例

```
{
  "BoxId": "84135354354355313132",
  "Status": "offline"
}
```

2.2.4.4 设备实时数据订阅示例

1、场景描述：监控设备实时运行数据，用于大屏展示和实时分析

（1）订阅主题：

/clean/84135354354355313132/realtime (QoS: 0)（监控设备 84135354354355313132 的实时消息）

/clean/+/realtime （监控所有设备的实时消息）

（2）预期接收消息示例

```
{
  "identifier": "1962403053254430720",
  "properties": [
    {
      "dataId": "1962403053342511104",
      "dataType": "2",
      "ts": "1759135076345",
      "value": "25.5"
    },
    {
      "dataId": "1962403053342511105",
      "dataType": "2",
      "ts": "1759135076345",
      "value": "65.2"
    }
  ],
```

```
    "tenantId": "HW6000001176646749",
    "type": "device"
  }
```

2.2.4.5 控制结果订阅示例

1、场景描述：监控反向控制指令的执行结果

（1）订阅主题：

/commService/boxDatas/resp/84135354354355313132 (QoS: 2)

/commService/boxDatas/resp/+ (QoS: 2) - 所有设备

（2）预期接收消息

```
{
    "result": 0,
    "param": {
      "type": "writeData"
    }}
```

第三章 系统对接

3.1 场景介绍

本章详细阐述了本系统与外部系统、设备、服务进行集成的能力与方式。作为二次开发人员，您可以通过本章了解系统现有的对接框架、接口规范和数据格式，并在此基础上进行扩展开发，实现定制化的集成需求。

系统对接主要围绕以下几个核心维度展开：

- 与物理设备的对接：通过物模型标准化设备接入与数据交互。
- 与数据流的对接：实现实时数据的推送与订阅，以及历史数据的查询与同步。
- 与数据存储的对接：直接与数据库进行交互，进行深度数据操作。
- 与业务系统的对接：与其他应用系统进行交互。

3.2 物模型对接

通过物模型对接可以实现自定义设备接入与数据交互，接入协议只支持 mqtt 接入。下表是其交互通道定义：

名称	主题	描述
实时数据上报	/${id}/real/data/default	实时上报点表数值

设备状态变更	/${id}/real/notify/status	发布设备、点表状态
设备注册	/box/info/${tenantId}/${Id}	设备注册添加
设备点表上报	/box/config/${tenantId}/${id}	设备点表动态配置

1、实时数据

```
{
    "boxId": "1100022112110000041",
    "realData": [
        {
            "deviceId": 1,
            "variables": [
                {
                    "dId": 1,
                    "dVal": 0,
                    "ts": 1733377290152
                },
                {
                    "dId": 2,
                    "dVal": 0,
                    "ts": 1733377290152
                }
            ]
        }
    ]
}
```

2、实时状态

```
{
    "notiyStatus": [
        {
```

```
            "rtDataTopic": "/3300224010215080061/real/data/default",
            "devices": [
                {
                    "deviceId": 15,
                    "deviceName": "烟雾感应器 1",
                    "status": "online",
                    "variableLists": [
                        {
                            "dId": 32,
                            "dName": "烟雾报警",
                             "status": "online"
                        }
                    ]
                }
            ]
        }
    ]
}
```


3、基础信息


```
{
    "tenantId": "HW01010012",
    "boxId": "网关编号",
    "userId": "用户编号",
    "boxName": "网关名称",
    "boxType": "Host4G",
    "versionType": 1,
    "operateType": "操作类型，-1：删除，0：更新，1：新增"
```

```
}
```

4、网关配置同步

```
{
    "profile": {
        "tenantId": "HW01010012",
        "userId": "1231"
    },
    "basicInfo": {
        "boxId": "网关编号",
        "boxName": "设备名称",
        "source": "BoxPlugin",
        "operationType": "update",
        "subDevices": [
            {
                "deviceId": 1,
                "deviceName": "TCP",
                "portName": "lan",
                "variableData": [
                    {
                        "type": "collect",
                        "deviceId": 2,
                        "dataId": "变量标识",
                        "name": "变量名称",
                        "optionType": "r",
                        "dataType": "bool，支持 bool、int、float、double、string",
                        "decimal": 2                    }
                ]
            }
```

```
        ]
    },
    "events": []
}
```

3.3 实时数据对接

设备实时数据

设备实时数据采用 Mqtt 消息完成数据分发。主题中${deviceId}为设备标识

名称	实时变量数据
Topic	/clean/${deviceId}/realtime
发布/订阅	发布
QoS	0
Payload	{ "identifier": "12001301299192101", "type": "gateway、device、site", "profile": { "tenantId": "HW01010012" }, "properties": [{ "dataId": "12", "value": "123.21", "dataType": "double", "ts": "1601238176763" }] }
备注	Identifier:标识符，系统唯一 type：类型，gateway：网关、device：设备、site：站点 profile：用户信息 tenantId：租户 id userId：用户 id dataId: 变量标识, value: 数值，根据计算规则改变数值, ts: 时间戳，该时间戳由转换实时数据中获取，不做任何改变

3.4 历史数据对接

接口名称	设备历史数据查询
功能描述	根据条件查询变量历史记录
接口地址	/api/hiwoo-iot/v1/equipment/history/list-by-time
请求方式	GET

请求参数：

Header 参数

header	必选	类型	说明
satoken	是	string	token

请求字段

字段	必选	类型	说明
variableId	是	String	变量标识
startTime	否	String	开始时间（为空查所有）格式：YYYY-MM-DD HH:mm:SS
endTime	否	String	结束时间（为空查所有）格式：YYYY-MM-DD HH:mm:SS

返回响应：

```
{
    "code": 200,
    "data": {
        "total": "总计",
        "size": "每页大小",
        "current": "当前页",
        "dataList": [
            {
                "dataId": "数据标识",
                "time": "采集时间，13 位精确到毫秒",
                "value": "采集值"
            }
        ]
    },
    "msg": ""
    }
```

3.5 数据库对接

3.5.1、应用场景

Things 平台通过开放数据库对接能力，支持用户直接访问平台数据，深度挖掘数据价值，并灵活对接各类第三方平台与应用，满足数据分析、可视化展示、智能决策等多样化业务需求。

典型应用场景包括：

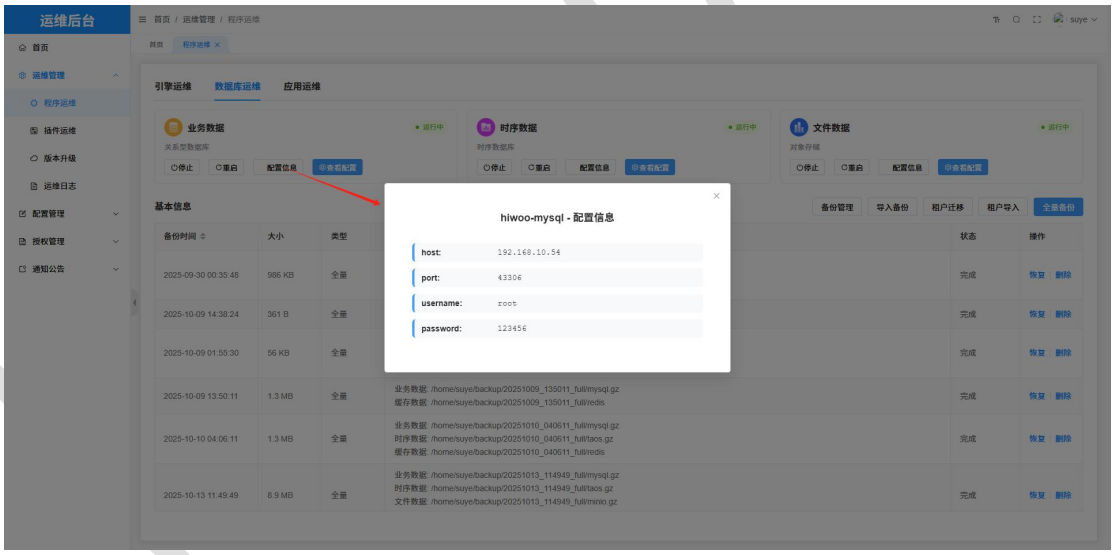
关系型数据库：用于管理结构化的、以实体和关系为核心的数据。其典型场景是各类需要保证数据一致性、完整性和支持复杂查询的业务系统。

时序数据库：用于高效地收集、存储和查询按时间顺序产生的数据序列。其核心是处理带时间戳的指标数据。

3.5.2、数据库开放

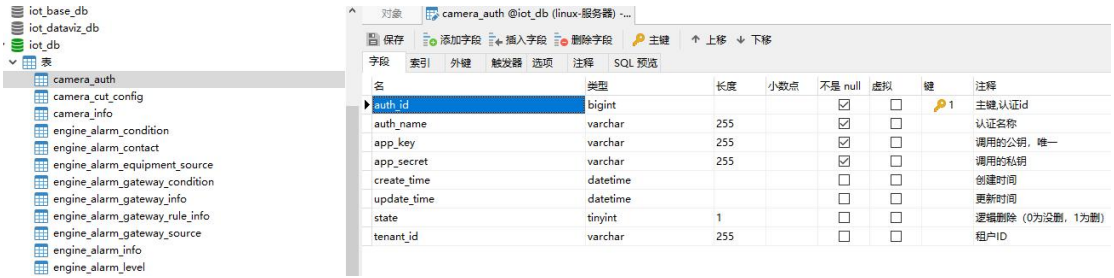
3.5.2.1、关系型数据库

- **连接参数获取**



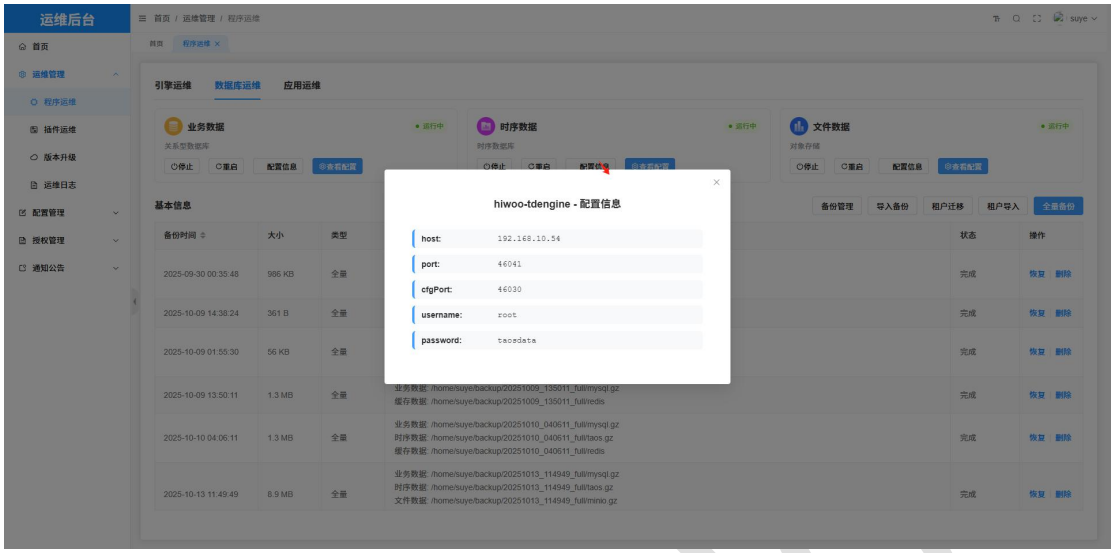
- 表结构查看

随意使用 Navicat 等常见数据库查看表结构、字段解释等信息



3.5.2.2、时序数据库

- 连接参数



点击配置信息按钮，弹出配置连接信息

3.6 用户系统对接

3.6.1、应用场景

Things 平台用户部分通过开放凭证授权能力，支持用户在保证账号安全的同时也能授权二开，您在实现对平台数据挖掘和二次应用场景时，可通过凭证授权后进行对平台提供的一系列二开接口进行授权调用。系统部分通过开放平台消息、日志与项目对接能力，如果您有多个系统，支持您以 Things 为中心进行二开使得数据与平台的深度融合。

3.6.2、Open API 接口开放

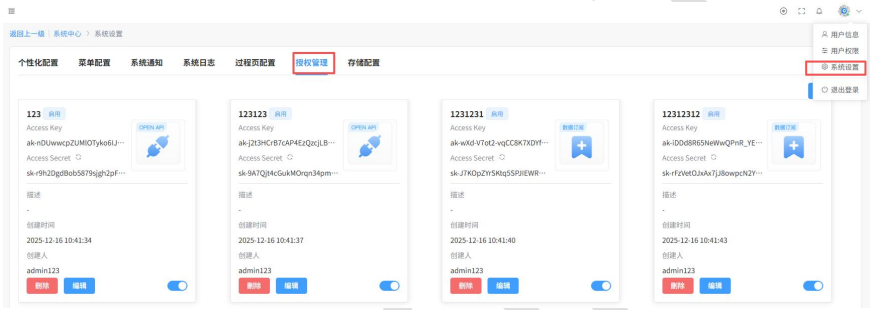
3.6.2.1、access 凭证认证

使用方式	第三方平台数据对接凭证
接口地址	/api/hiwoo-base/uaa/v1/auth/access
请求方法	POST
请求类型	application/json

请求体	{
	"accessKey": "您的凭证公钥",
	"accessSecret": "您的凭证私钥",
	}

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760443856385,
	"data": "您的访问凭证"
	}

失败示例	{
	"code": 401,
	"time": 1760443856385,
	"msg": "您的凭证无效"
	}

备注	平台申请凭证参考下图示例： 

3.6.2.2、系统日志同步

使用方式	第三方平台或插件日志生成
接口地址	/api/hiwoo-base/logs/v1/create
请求方法	POST
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求体	{
	"type": "日志类型",
	"userName": "人员名称",
	"content": "日志内容",
	}

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760443856385,
	"data": "true"
	}

失败示例	{
	"code": 500,
	"time": 1760443856385,
	"msg": "处理失败"
	}

备注	类型：【1】登录日志；【2】操作日志；【3】设备日志

3.6.2.3、项目树（无资源）

使用方式	第三方平台或插件获取项目树结构
接口地址	/api/hiwoo-base/projects/v1/get/tree
请求方法	GET
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求体	无

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760527748213,
	"data": [
	{
	"id": "1958341377768022018",
	"name": "一厂区项目",
	"type": "custom",
	"pid": "0",
	"description": "~~~~~~",
	"isEnabled": "enable",
	"tenantId": "HW6000000127608400",
	"updateTime": "2025-08-21 09:32:23",
	"createTime": "2025-08-21 09:32:23",
	"tag": null,
	"orgId": "1939624641084821504",
	"orgName": "根组织",
	"children": []
	}
	]
	}

失败示例	{
	"code": 500,
	"time": 1760443856385,
	"msg": "处理失败"
	}

备注	

3.6.2.3、项目树（含资源）

使用方式	第三方平台或插件获取项目树结构形式的资源
接口地址	/api/hiwoo-base/projects/v1/get/tree/{resourceTag}
请求方法	GET
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求参数	"resourceTag": "资源类型",
成功示例	{ "code": 200, "msg": "处理成功", "time": 1760528344022, "data": [{ "projectId": "1958341377768022018", "applicationTag": null, "resourceTag": null, "resourceId": null, "resourceName": null, "pid": "0", "projectName": "tesst", "children": [{ "projectId": "1958341377768022018", "applicationTag": "iot", "resourceTag": "equipment", "resourceId": "1978010869414367232", "resourceName": "3123121", "pid": "1958341377768022018", "projectName": null, "children": [

	{
	"projectId": "1958341377768022018",
	"applicationTag": "iot",
	"resourceTag": "variable",
	"resourceId": "1978013283253178369",
	"resourceName": "1123",
	"pid": "1978010869414367232",
	"projectName": null,
	"children": null
	}
	]
	}
	]
	}
	]
	}

失败示例	{
	"code": 500,
	"time": 1760443856385,
	"msg": "处理失败"
	}

备注	系统资源标签：【screen】综合大屏；【screen-template】大屏模板；【dashboard】工艺看板；【dashboard-template】看板模板、【analysis】常规报表；【advanced】高级报表；【equipment】物联设备；【alarm】平台报警；【gateway-alarm】网关报警；【contact】报警联系人；【media】媒体视频 自定义资源标签：参考您插件注册资源前定义的“resource-tag”属性

3.6.3、MQTT 主题开放

3.6.3.1、通知消息订阅

名称	通知消息同步
Topic	/sync/notice
发布/订阅	发布
QoS	1
Payload	{ title:"消息标题", type:"消息类型",

	content:"消息内容", releaseTime:"推送时间", receiverType:"接收群体", receiver:"用户标识", //非必填 tenantId:"租户标识", //非必填 method:"通知方式" //多个方式采用“,”连接符分隔 }
备注	type：【systemMsg】公告消息；【alarmMsg】报警消息；【eliminateMsg】报警消除消息 receiverType：【all】全员；【tenant】租户；【user】用户 method：【platform】平台；【sms】短信；【email】邮箱

3.6.3.2、示例参考

场景描述：需要同步消息到平台

（1）订阅主题：/sync/notice

QoS：1

（2）订阅消息接收：

```
{
    title:"全站公告",
    type:"systemMsg",
    content:"将于 2025 年 12 月 31 日进行平台运维，届时平台将停止运行~",
    releaseTime:"2025-12-20 12:00:00",
    receiverType:"all",
    receiver:"",
    tenantId:"",
    method:"platform,sms"
}
```

第四章 插件开发

4.1、应用场景

插件开发是 Things 平台二次开发的核心能力之一，适用于多种业务集成与功能扩展场景。通过插件机制，用户可以将自研应用、第三方服务或定制化功能无缝集成到 Things 平台中，实现业务逻辑的灵活扩展与系统能力的增强。

典型应用场景包括：

功能扩展：当平台标准功能无法满足特定业务需求时，可通过开发自定义插件实现功能增强，如定制报表、专用设备协议解析、业务规则引擎等。

第三方系统集成：通过插件形式接入企业已有的业务系统（如 ERP、MES、CRM 等），实现数据同步、业务流程打通或界面集成。

定制化业务逻辑：针对特定行业或场景，开发具备独立业务处理能力的插件，如智能告警、数据分析看板、自动化控制策略等。

数据对接与处理：开发专门的数据处理插件，对接外部数据源（如传感器数据、视频流、第三方 API 等），并进行清洗、转换、存储或转发。

界面定制：通过前端插件扩展平台界面，嵌入自定义页面、图表或操作面板，提升用户体验与操作效率。

插件系统支持灵活部署方式，用户可选择使用**系统插件**（由合沃官方提供并授权）或**自定义插件**（用户自行开发），满足从标准化功能到完全定制化的各类业务需求。

4.2、插件管理

插件管理是指通过运维后台构建、管理插件的系统，以下是其使用说明:

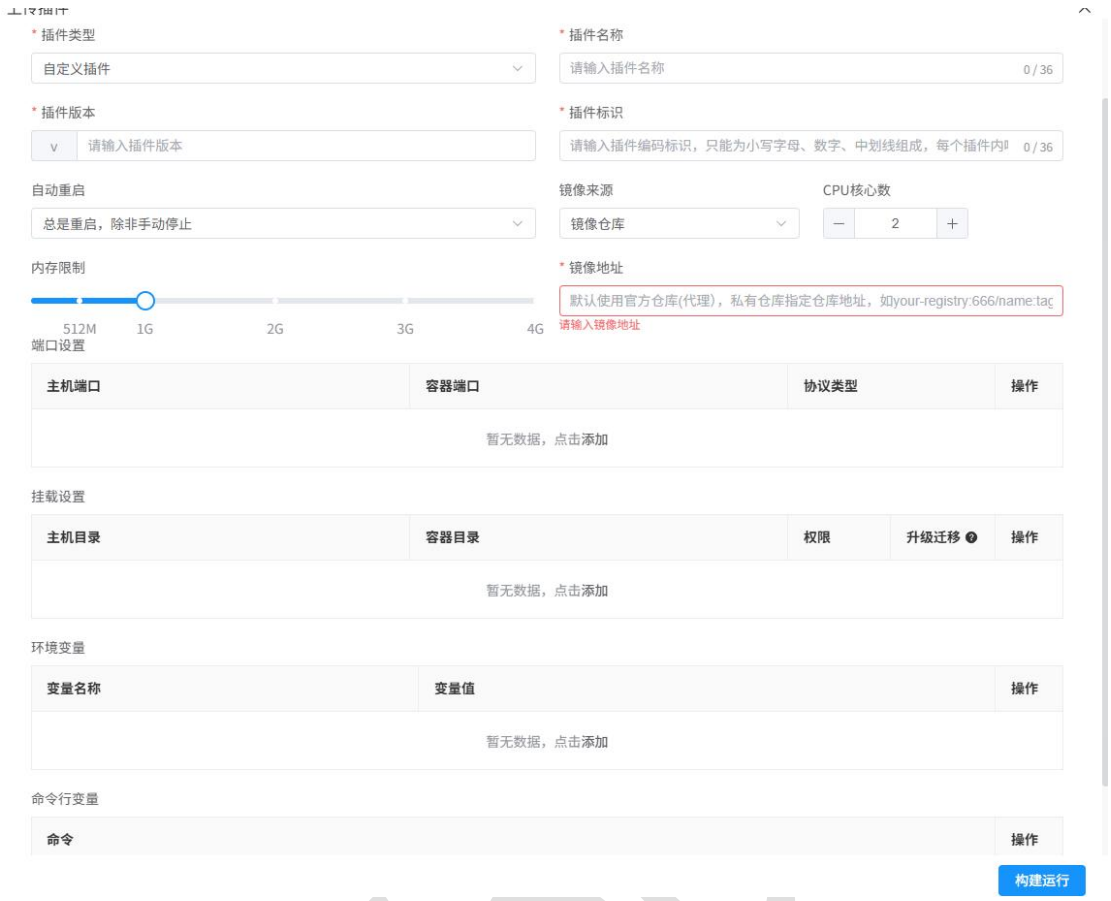
4.2.1、插件类型

- 系统插件

由合沃官方出品的插件，需授权。选择对应插件包后插件配置会自动设置，也可根据实际情况进行微调。

- 自定义插件

由用户上传提供的插件。可选择离线部署包和镜像仓库两种方式。同时可根据需要配置端口、挂载、环境变量、命令参数等启动配置



4.2.2、镜像来源

离线部署包：一个 tar.gz 文件。需要符合一定规范，后续详解

镜像仓库：指定官方镜像仓库地址或三方仓库镜像地址。即同 docker pull 命令指定的镜像包。

4.2.3、离线插件包

资源	是否必须	命名规范	类型	放置目录	说明
插件镜像包	是	Dockerfile app.tar	文件	根目录	二选一，必须提供。如果都提供，则使用 Dockerfile 构建
依赖镜像包	否	images	文件夹	根目录	只会导入该文件夹下的.tar 文件

Nginx 代理配置	否	nginx.conf	文件	根目录	
升级脚本	否	[before/after]-[remove/install].sh	文件	根目录	只在升级时执行
其他文件	否	无	任意	任意	插件自行选择提供的文件，任意

4.2.4、插件镜像包

支持通过 Dockerfile 构建或者直接提供导出的镜像（tar 文件）。DockerFile 文件或镜像包都需放在根目录。Dockerfile 文件名只可为 Dockerfile。镜像包名称必须为 app.tar 才可识别。

4.2.5、依赖镜像包

如您的插件需要其他依赖镜像包，如 java 项目一般需要通过 Jre 镜像构建，这个 jre 镜像就可放置在插件根目录下的 images 文件夹下，插件部署时会提前将这些镜像导入到系统中。

3.3、nginx 反向代理

如您的插件为一个 web 项目，插件也提供了 nginx 反向代理能力。您可以将 nginx 配置文件命名为 nginx.conf，放置在根目录。插件会设别并进行代理设置。

```
location /plugin-api/ {
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-Proto https;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_redirect off;
        proxy_connect_timeout 240;
        proxy_send_timeout 240;
        proxy_read_timeout 240;
        proxy_pass http://hiwoo-app-host:15555/;
```

```
 }


location /plugin-test/ {
        add_header Access-Control-Allow-Origin *;
        add_header Access-Control-Allow-Headers X-Requested-With;
        add_header Access-Control-Allow-Methods GET,POST,PUT,DELETE,PATCH,OPTIONS;
        add_header Cache-Control "no-store, no-cache, must-revalidate";

        try_files $uri $uri/ /index.html;
        alias $work_dir/html/;
        index index.html index.htm;
}
```

nginx 配置文件只能包含 location 块，但内容由您完全自定义。同时插件系统会为您注入几个参数，您可以按需求使用：

$work_dir: 代表插件的工作目录。即您插件包的根目录。如上 alias $work_dir/html/ 表示在插件包中有一个名为 html 文件夹的静态资源包，通过/plugin-test 代理访问该资源。

hiwoo-app-host:host,代表部署服务器的主机 ip

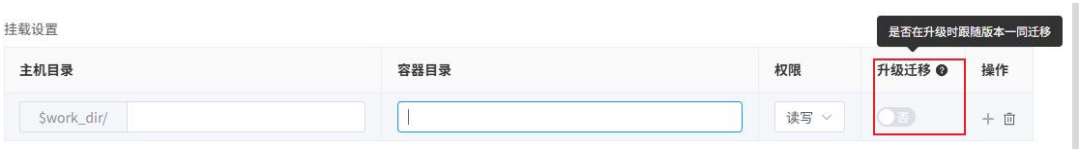
hiwoo-data-host:host,混合部署下代表数据服务器的主机 ip。单机部署即为主机 ip 同 hiwoo-app-host

4.2.6、升级迁移

为了方便插件进行平滑升级迁移。插件系统提供了几个钩子函数

脚本名称	执行时机	提供参数
before-remove.sh	原插件删除前执行	第一个参数$1：old_version 第一个参数$2：new_version
after-remove.sh	原插件删除后执行	
before-install.sh	新插件安装前执行	
after-install.sh	新插件安装完成后执行	

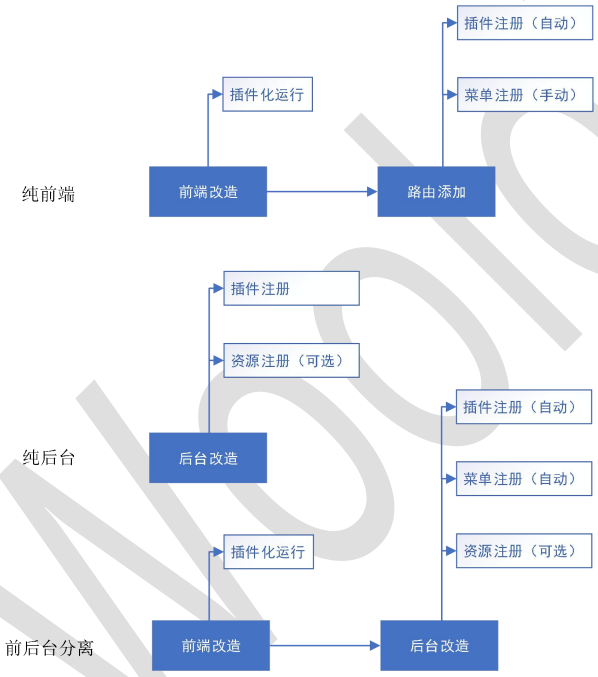
特别说明：插件脚本执行环境为虚拟环境，您仅可以操作原插件安装目录与新插件安装目录。两这分别被挂载到/app/old、/app/new 目录，您可以通过该位置访问插件包里的具体文件



这里特别说明下升级迁移开关，其作用为简单的升级数据迁移。如您的插件存在一个文件上传目录，则可以将此开关打开。升级时会将原插件的上传目录拷贝至新插件的工作目录。

4.3、插件融合

插件融合分为纯前端插件、纯后台插件、前后台分离插件，各自融合方式参考如下步骤图：



4.3.1、前端插件改造

本章节以主流前端框架组合 Vue3+Element plus+Vite4 为例，如下步骤完成工程具备插件加载的条件。

4.3.1.1、依赖安装

①npm 安装命令：

```
npm install vite-plugin-qiankun@1.0.15 --save-dev
```

主应用采用 qiankun 2.10.x 版本，所以此处建议子应用 vite-plugin-qiankun 依赖采用>=1.0.15 以上版本。此依赖兼容 vite 2.x、3.x、4.x，但推荐使用 4.x 版本。

4.3.1.2、Vite 配置（vite.config.ts）

①导入插件依赖：

```
import qiankun from 'vite-plugin-qiankun'
```

②基础配置：

```
base: `plugin-demo`    //  “plugin-demo”为项目名称，此处根据项目自定义。
```

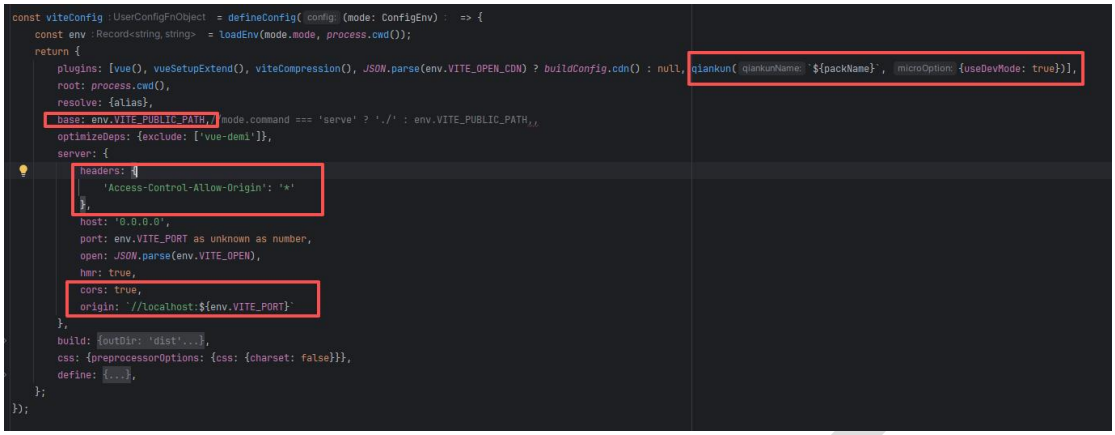
③Vite 插件拓展：

```
plugins: [
        ...（其他依赖）
        qiankun(`plugin-demo`, {useDevMode: true})    //  “plugin-demo”为插件名称，此处根据项目自定义。
]
```

④Vite 开发模式：

```
server:{
    headers: {
        'Access-Control-Allow-Origin': '*'
    },
    cors: true,
    origin: `//localhost:8888`,    //  “8888”为端口，此处根据项目自定义。
}
```

示例：



注意事项：

① 插件注册的名称要跟项目名称、部署包名称需要保持一致。项目名称不一致会导致开发模式下主应用找不到插件，部署包名称不一致会导致生成模式下静态访问主应用找不到插件。

②开发模式的 origin 必须要带上端口用于识别工程，插件加载时作为子应用视为主应用的一部分所以默认变成“localhost”访问，需要通过指向 IP 和端口的方式访问。否则会出现静态资源无法访问的情况

4.3.1.3、生命周期注册（main.ts）

①导入插件依赖：

```
import {renderWithQiankun, qiankunWindow} from 'vite-plugin-qiankun/dist/helper'
```

②运行挂载改造：

```
let app: any = null;
const render = ({routes, routerBase, container} = {}) => {
    app = createApp(App);
    /*
        ... （注入全局使用的组件）
    */
    // 挂载容器（检测插件加载走插件模式挂载，独立运行则走默认挂载）
    app.mount(qiankunWindow.__POWERED_BY_QIANKUN__?container.querySelector("#app")
```

```
      : '#app');
}
export const bootstrap = async (): Promise<void> => {
    /**
      * bootstrap 只会在微应用初始化的时候调用一次，下次微应用重新进入时会直接调
       用 mount 钩子，不会再重复触发 bootstrap。
      * 通常我们可以在这里做一些全局变量的初始化，比如不会在 unmount 阶段被销毁
        的应用级别的缓存等。
      */
}
export const mount = async (props: any): Promise<void> => {
  /**
    * 应用每次进入都会调用 mount 方法，通常我们在这里触发应用的渲染方法
    */
    // 广播消息
    const token = props.satoken
    const user = props.userInfo
    const globalI18n = props.globalI18n
    const theme = props.theme        //ElementPlus 可忽略（主题基于此 UI 封装）

    /*
        ... （根据业务自行接收处理广播消息）
    */

    // 全局状态数据监听
    props.onGlobalStateChange &&
    props.onGlobalStateChange(
        (value, prev) => console.log(`[onGlobalStateChange - ${props.name}]:`, value, prev),
        true
    )
```

```
    // 发送消息
    props.setGlobalState &&
    props.setGlobalState({
        ignore: props.name,
        user: {
            name: props.name
        }
    })
    render(props)
    app.config.globalProperties.$onGlobalStateChange = props.onGlobalStateChange
    app.config.globalProperties.$setGlobalState = props.setGlobalState
}
export const unmount = async (): Promise<void> => {
  /**
    * 应用每次 切出/卸载 会调用的方法，通常在这里我们会卸载微应用的应用实例
    */
    app.unmount()
    app._container.innerHTML = ''
    app = null
}

export const update = async (): Promise<void> => {
  /**
    * 可选生命周期钩子，仅使用 loadMicroApp 方式加载微应用时生效
    */
}

// 检测是否插件加载，决定加载模式
qiankunWindow.__POWERED_BY_QIANKUN__ ? renderWithQiankun({bootstrap, mount,
```

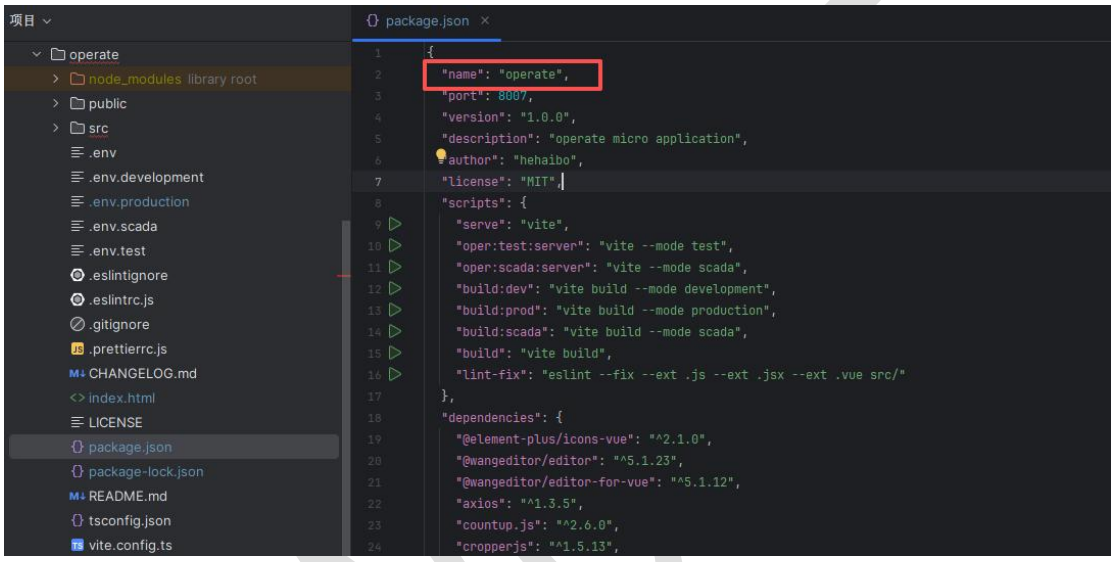
```
unmount, update}) : render();
```

4.3.1.4、history 路由模式（可选，hash 路由模式忽略）

①设置路由守卫

```
history: createWebHistory("/operate/")
```

这里建议统一以 package.json 中的 name，如下（示例工程名称为“operate”）：



当改造的工程路由为 history 模式，即可通过此方式设置。示例：

```
/**
 * 创建一个可以被 Vue 应用程序使用的路由实例
 * @method createRouter(options: RouterOptions): Router
 * @link 参考：https://next.router.vuejs.org/zh/api/#createrouter
 */
5+ 个用法  Haibo He
export const router :Router  = createRouter( options: {
    history: createWebHistory( base: "/operate/"),
    /**
     * 说明：
     * 1、notFoundAndNoPower 默认添加 404、401 界面，防止一直提示 No match found for location with path 'xxx'
     * 2、backEnd.ts(后端控制路由)、frontEnd.ts(前端控制路由) 中也需要加 notFoundAndNoPower 404、401 界面，
     *    防止 404、401 不在 layout 布局中，不设置的话，404、401 界面将全屏显示
     */
    routes: [...notFoundAndNoPower, ...staticRoutes],
});
```

注意事项：

① 基础路由名称要跟项目名称需要保持一致，否则主应用无法访问插件页面。

4.3.1.5、差异化加载（可选）

①导入插件依赖：

```
import {qiankunWindow} from 'vite-plugin-qiankun/dist/helper'
```

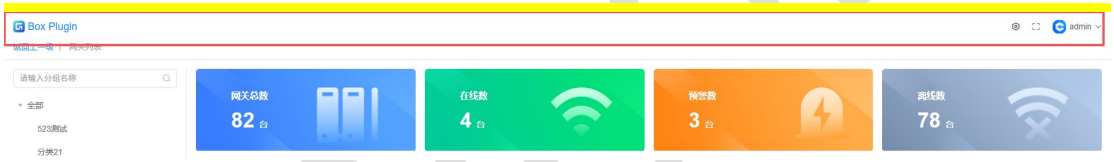
②差异化检测：

```
qiankunWindow.__POWERED_BY_QIANKUN__
```

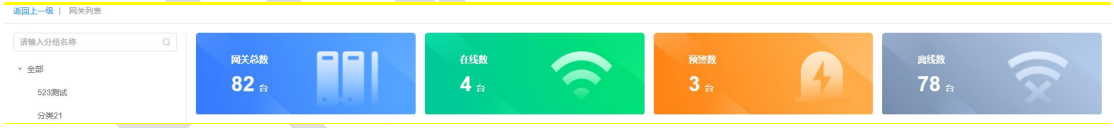
当希望改造的工程部分内容插件加载和独立运行具备且代码同源的情况下，即可通过此方式设置。示例：

```
<div v-if="!Session.get('noToolbar') &&  !qiankunWindow.__POWERED_BY_QIANKUN__"
     style="display: flex; flex-direction: row; align-items: center; justify-content: space-between; width: 100%;height: 50px;">
  <Logo/>
  <User/>
</div>
<Breadcrumb/>
```

改造前：



改造后：



4.3.1.6、SessionStorage 隔离（可选）

①为避免主子应用缓存机制相互污染，插件建议以插件名字为前缀存储数据

当改造的工程缓存机制处理时，即可通过此方式设置（与实际工程为例）。示例：

```
setKey(key: string) :string  {
    // @ts-ignore
    return `${require('./package').name}:${key}`;
},
// 设置永久缓存
set<T>(key: string, val: T) :void  {
    window.localStorage.setItem(Local.setKey(key), JSON.stringify(val));
},
// 获取永久缓存
get(key: string) {
    let json :string  = <string>window.localStorage.getItem(Local.setKey(key));
    return JSON.parse(json);
},
// 移除永久缓存
remove(key: string) :void  {
    window.localStorage.removeItem(Local.setKey(key));
},
```

4.3.2、前端数据拉通

本章为插件前端数据共享说明，如下所示将二开的插件可以监听获取主应用共享的数据信息，保证插件数据跟主应用同步。

4.3.2.1、认证信息

➢ 用户信息

接收方式	const user = props.userInfo	
消息格式	{	
	"id":"199*****",	
	"name":"admin123",	//用户名
	"headSculpture":null,	//头像
	"account":"admin123",	//账号
	"roleId":"1",	//角色编号
	"departmentId":"199*****",	//组织编号
	"enterpriseId":"HW*****",	//租户标识
	"phone":null,	//手机号码
	"email":null,	//邮箱

	"notes":null,
	"isLocked":0,
	"companyName":null,
	"openid":null,
	"miniProgramsOpenid":null,
	"thirdPartyMiniProgramsOpenid":null,
	"thirdPartyOpenid":null
	}
		//个性签名
		//是否锁号；0：正常，1：锁号
		//租户企业名称
		//微信公众号 ID
		//微信小程序 ID
		//授权三方微信公众号 ID
		//授权三方微信小程序 ID

备注说明	主应用加载子应用时会广播消息，所以子应用在 mount 时处理接收。	

➢ 授权 token

接收方式	const token = props.satoken
消息格式	"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJsb2dpblR5cGUiOiJsb2dpbiIsImxvZ2luSWQiOiJhZG1pbjEyMyIsInJuU3RyIjoiZmpLU1BSeFNPOXVvYzdzWFFObXBwRUd2dWFmVnlWeFIiLCJ0ZW5hbnRJZCI6IkhXNjAwMDAwMTU0OTU0MjE5MyIsInVzZXJJZCI6MTk5NjE0MjY4NzA4NjkxNTU4NSwicm9sZUlkIjoiMSIsImRlcGFydG1lbnRJZCI6IjE5OTYxNDI2ODY4ODk3ODMyOTgifQ"
备注说明	主应用加载子应用时会广播消息，所以子应用在 mount 时处理接收。

4.3.2.2、系统信息

➢ 系统语言

接收方式	const token = props.globalI18n	
消息格式	"zh-Cn"	//当前系统支持 2 种语言； 【zh-Cn】中文；【en】英文
备注说明	主应用加载子应用时会广播消息，所以子应用在 mount 时处理接收。	

➢ 系统样式

接收方式	const theme = props.theme

消息格式	{
	"theme":"white",
	"style":{
	text-color:"#FFF***",
	background-color:"#FFF***",
	},
	}
		//当前系统支持 4 种主题；
		【white】简约白；【green】环保绿；
		【blue】科技蓝；【orange】能源橙
		
		如果您的项目工程用的 Element plus 的 UI 可直接忽略,主应用基于此 UI 封装的主题。

备注说明	主应用加载子应用时会广播消息，所以子应用在 mount 时处理接收。	

➢ 系统菜单

接收方式	const menus = props.oldRoutes	

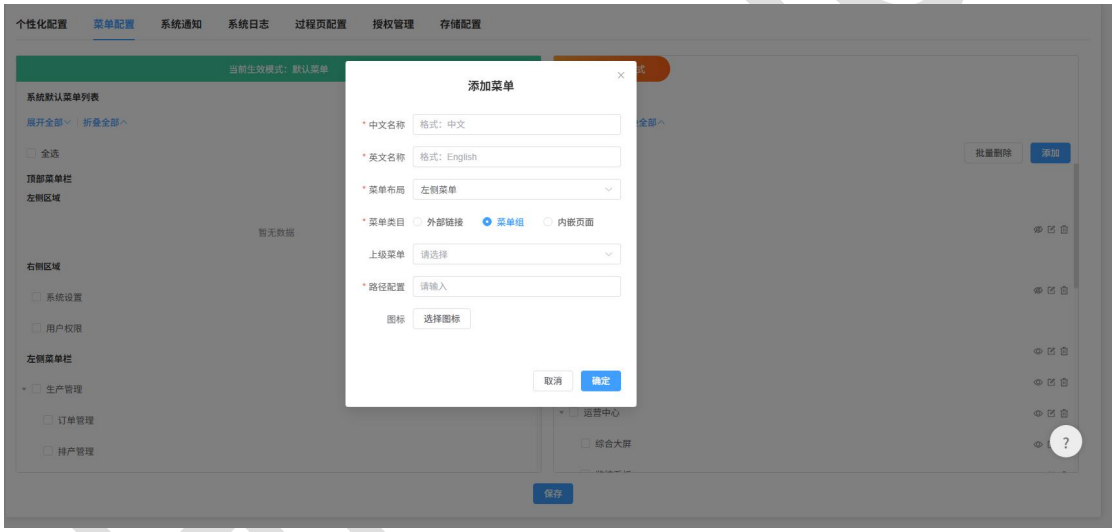
消息格式	[
	{
	"id": 0,
	"pluginId": "",
	"pid": 0,
	"path": "",
	"entry": "",
	"activeRule": "/operate",
	"menuType": "M",
	"title": "",
	"icon": "Monitor",
	"menuOrder": 1,
	"zhCn": "运营中心",
	"en": "Operate Center",
	"applicationTag": "",
	"menuTag": "",
	"keepAlive": true,
	"affix": false,
		
		//插件名称
		//菜单从属
		//菜单路由
		//插件来源
		//插件路由
		//菜单类型；M：菜单组，C：菜单
		//弃用
		//菜单 ICON
		//菜单排序
		//菜单名称（中文）
		//菜单名称（英文）
		//插件标签
		//菜单标签
		//是否缓存组件状态
		//弃用

	"iframe": false,
	"link": false,
	"hide": false,
	"children": [],
	"prem": ["view","edit"],
	}
	]
		//是否内嵌窗口
		//弃用
		//是否隐藏显示
		//子菜单
		//菜单操作权限

备注说明	主应用加载子应用时会广播消息，所以子应用在 mount 时处理接收。	

4.3.3、前端插件注册

本章为纯前端插件注册配置说明，如下所示将二开的前端插件以手动配置的方式添加到系统菜单路由，平台主应用根据路由自动注册访问插件页面。



4.3.4、后台插件注册

本章为纯后台插件和前后台分离插件注册说明，如下所示将二开的插件以自动注册的方式添加到系统菜单路由，平台主应用根据注册路由访问插件页面。

4.3.4.1、消息主题总览

主题分类	主题名	主题	QoS	是否保留
插件注册订阅	插件注册	/register/application/{applicationTag}	1	false
	资源注册	/register/application/resource/{resourceTag}	1	false

4.3.4.2、插件注册主题

名称	插件注册及相关资源类目和资源操作类型注册
Topic	/register/application/{applicationTag}
发布/订阅	发布
QoS	1

Payload	{
	applicationName:"插件名称",
	applicationTag:"唯一标识", //建议使用服务名称
	applicationType:"应用类型",
	resources:[
	{
	resourceTag:"资源标签", //您按资源自定义
	zhCn:"资源名称",
	en:"资源名称（英文）",
	resources:[二级资源] //如：设备下的变量等
	operates:[
	{
	operateTag:"操作标签",
	zhCn:"操作名称",
	en:"操作名称（英文）",
	}
	]
	}
	]
	}

备注	applicationType：【application】平台应用；【plugin】系统插件； 【customPlugin】自定义插件

名称	插件的资源注册/变更时同步
Topic	/register/application/resource/{resourceTag}
发布/订阅	发布
QoS	1
Payload	{ applicationName:"插件名称", applicationTag:"唯一标识", //建议使用服务名称 applicationType:"应用类型",

	resources:[
	{
	resourceTag:"资源标签", //各个应用按资源自定义
	zhCn:"资源名称",
	en:"资源名称（英文）",
	resources:[二级资源] //如：设备下的变量等
	operates:[
	{
	operateTag:"操作标签",
	zhCn:"操作名称",
	en:"操作名称（英文）",
	}
	]
	}
	]
	}

备注	opreation：【create】新增资源；【edit】编辑资源；【delete】移除资源

4.3.4.3、插件注册示例

1、**场景描述**：需要注册数据可视化（applicationTag：operate）的插件

（1）订阅主题：**/register/application/operate**

QoS：1

（2）订阅消息接收：

```
{
   applicationName:"数据可视化插件",
   applicationTag:"operate",
   applicationType:"customPlugin",
   resources:[
      {
         resourceTag:"screen",
         zhCn:"综合大屏",
         en:"Integrated Dashboard",
         operates:[
             {
                 operateTag:"view",
                 zhCn:"查看",
                 en:"view",
             }
         ]
          resources:[]     // 如有二级资源递归填充
      }
```

```
    ]
}
```

2、场景描述：需要同步数据可视化插件下大屏资源的数据（resourceId：12354355313132）

（1）订阅主题：/register/application/resource/operate

QoS：1

（2）预期接收消息示例

```
{
    opreation:"create",
    applicationTag:"operate",
    resourceTag:"screen",
    resourceId:"12354355313132",
    resourceName:"光伏电站能源管理大屏",
    projectId:"123456789101112",
    chlidren:[]    // 如有二级资源递归填充
}
```

4.4、引擎开放

4.4.1 定时任务引擎 xxl-job

XXL-JOB 是一款分布式任务调度平台，为企业提供统一的定时任务管理和调度服务。

4.4.1.1 服务器连接信息

调度中心（管理后台）

访问地址：http://ip:端口/xxl-job-admin

默认账号：admin/123456

OpenAPI 地址：http://ip:端口/xxl-job-admin/api/

认证令牌：default_token

4.4.1.2 核心应用场景

（1）跨系统数据同步

① ERP/MES 数据交互：定时同步生产数据、库存信息

② 第三方系统对接：与供应商、客户系统的数据交换

③ 数据库间数据同步：不同业务库之间的数据一致性维护

（2）业务统计分析

① 定时报表生成：每日/每周/月度业务报表

② 数据聚合计算：销售统计、用户行为分析

③ 业务指标监控：KPI 指标计算与预警

（3）设备与资源管理

① 设备定时控制：设备开关机、模式切换

② 数据备份任务：定时备份数据库、文件数据

③ 资源清理维护：日志清理、缓存刷新、临时文件删除

4.4.1.3 管理后台操作

步骤 1：登录管理后台

访问：http://ip:端口/xxl-job-admin

账号：admin/123456

步骤 2：验证执行器状态

进入"执行器管理"

查看执行器 "hiwoo" 是否在线

确认注册地址为：http://hiwoo-iot:35555

步骤 3：创建定时任务

进入"任务管理"

点击"新增"

填写任务配置信息

4.4.1.4 常用 API 接口

功能	接口地址	方法	说明
任务列表	/jobinfo/pageList	GET	分页查询任务
添加任务	/jobinfo	POST	创建新任务
更新任务	/jobinfo	PUT	修改任务配置

功能	接口地址	方法	说明
删除任务	/jobinfo/{id}	DELETE	删除任务
触发执行	/jobinfo/trigger	POST	立即执行任务
任务日志	/joblog/pageList	GET	查询执行日志

4.4.1.5 调度规则配置

（1） Cron 表达式示例

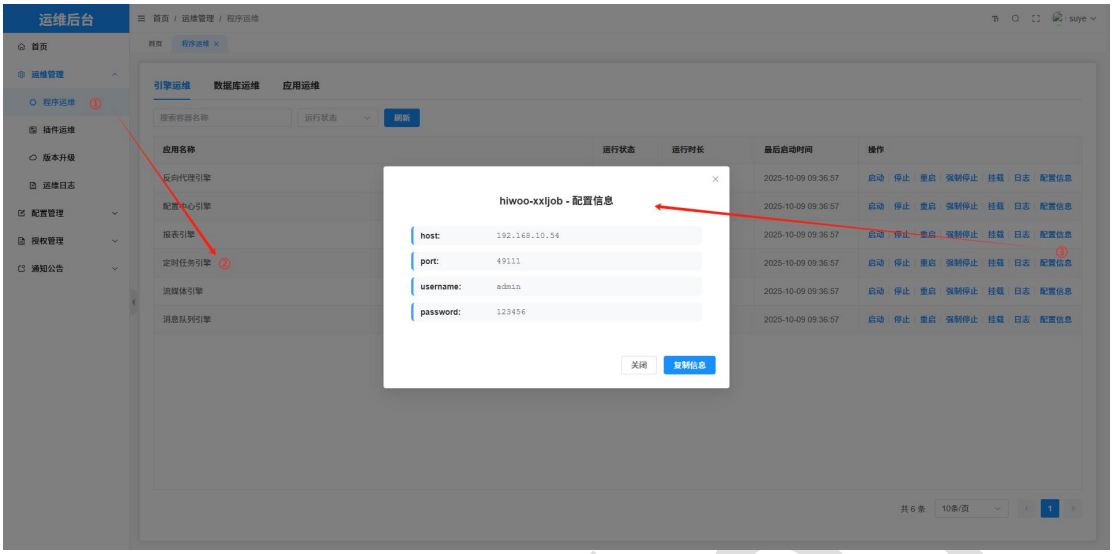
```
{
  "scheduleType": "CRON",
  "scheduleConf": "0 0 2 * * ?",      // 每天凌晨2点
  "scheduleConf": "0 0/30 * * * ?",   // 每30分钟
  "scheduleConf": "0 0 9-18 * * ?",   // 每天9点到18点整点
  "scheduleConf": "0 0 12 * * 1-5"    // 工作日中午12点
}
```

（2）固定定频率示例

```
{
  "scheduleType": "FIX_RATE",
  "scheduleConf": "300",      // 每5分钟
  "scheduleConf": "3600",     // 每1小时
  "scheduleConf": "86400"     // 每24小时
}
```

【注】详细使用请参考 xxl-job 官方使用文档

4.4.1.6 获取连接配置



点击定时任务引擎的配置信息按钮，即可弹出弹框显示配置连接信息

4.4.2 流媒体引擎 zlmediakit

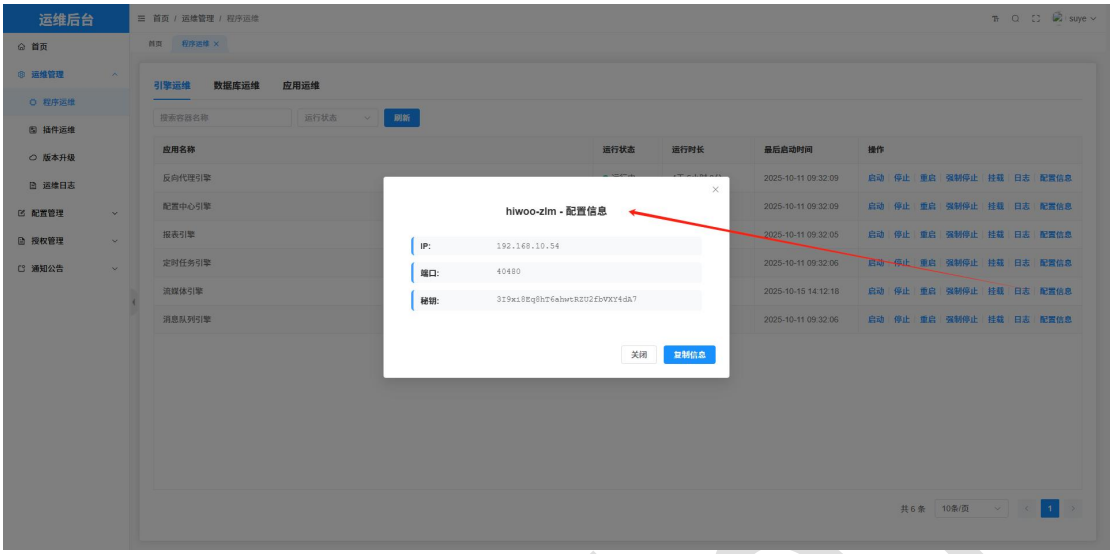
ZLMediaKit接口文档.md

zlmediakit 提供 API，可以用于控制和管理 ZLMediaKit 流媒体服务器的功能和状态。

4.4.2.1 基础连接信息

服务器地址：http://IP:端口
API 路径：/index/api/
认证密钥：3I9xi8Eq8hT6ahwtRZU2fbVXY4dA7
2、流媒体接口：40480

4.4.2.2 配置连接信息获取



点击流媒体引擎的配置信息按钮，即可弹出弹框显示配置连接信息。

4.4.2.3 核心功能使用指南

（1） 服务器状态监控

检查服务器状态：

```
# 获取服务器配置
curl "http://your-server:port/index/api/getServerConfig?secret=your_secret"
```

（2）视频流管理

添加拉流代理（从摄像头拉取视频流）

```
curl "http://your-server:port/index/api/addStreamProxy?\
secret=your_secret&\
vhost=__defaultVhost__&\
app=live&\
stream=camera001&\
url=rtsp://admin:password@192.168.1.100:554/stream1&\
enable_hls=1&\
enable_mp4=1"
```

参数说明：

① vhost: 虚拟主机（通常使用 __defaultVhost__）

② app: 应用名称（如 live）

③ stream: 流 ID（自定义，如 camera001）

④ url: 源视频流地址

⑤ enable_hls: 是否生成 HLS 流（1=是，0=否）

⑥ enable_mp4: 是否录制 MP4（1=是，0=否）

【注】在这里**流 ID** 可以进行自定义，流 ID 的推荐命名格式：stream_id = "位置_设备类型_编号"

```
# 示例：
camera_office_001        # 办公室 1 号摄像头
camera_parking_002       # 停车场 2 号摄像头
device_door_001          # 门禁 1 号设备
```

（3） 检查流状态

```
curl "http://your-server:port/index/api/isMediaOnline?\
secret=your_secret&\
schema=rtsp&\
vhost=__defaultVhost__&\
app=live&\
stream=camera001"
```

（4）关闭视频流

```
curl "http://your-server:port/index/api/close_stream?\
secret=your_secret&\
schema=rtsp&\
vhost=__defaultVhost__&\
app=live&\
stream=camera001&\
force=1"
```

（5）录制管理

```
# 开始 MP4 录制 curl "http://your-server:port/index/api/startRecord?\
secret=your_secret&\
type=1&\
```

```
vhost=__defaultVhost__&\
app=live&\
stream=camera001"
```

（6）停止录制

```
curl "http://your-server:port/index/api/stopRecord?\
secret=your_secret&\
type=1&\
vhost=__defaultVhost__&\
app=live&\
stream=camera001"
```

（7）检查录制状态

```
curl "http://your-server:port/index/api/isRecording?\
secret=your_secret&\
type=1&\
vhost=__defaultVhost__&\
app=live&\
stream=camera001"
```

（8）获取流列表

```
curl "http://your-server:port/index/api/getMediaList?\
secret=your_secret&\
app=live"
```

4.4.2.4 播放地址说明

成功添加流后，可以使用以下地址播放：

HLS 播放：

```
http://your-server:port/live/camera001/hls.m3u8
```

FLV 播放：

```
http://your-server:port/live/camera001.flv
```

RTMP 播放：

```
rtmp://your-server:port/live/camera001
```

4.4.2.5 常用摄像头流地址

1、海康威视

rtsp://[username]:[password]@[ip]:[port]/[codec]/[channel]/[subtype]/av_stream

说明：

username: 用户名。例如 admin。

password: 密码。例如 12345。

ip: 为设备 IP。例如 192.0.0.64。

port: 端口号默认为 554，若为默认可不填写。

codec：有 h264、MPEG-4、mpeg4 这几种。

channel: 通道号，起始为 1。例如通道 1，则为 ch1。

subtype: 码流类型，主码流为 main，辅码流为 sub。

例如，请求海康摄像机通道 1 的主码流，Url 如下

主码流：

rtsp://admin:12345@192.0.0.64:554/h264/ch1/main/av_stream

rtsp://admin:12345@192.0.0.64:554/MPEG-4/ch1/main/av_stream

子码流：

rtsp://admin:12345@192.0.0.64/mpeg4/ch1/sub/av_stream

rtsp://admin:12345@192.0.0.64/h264/ch1/sub/av_stream

2、大华

rtsp://[username]:[password]@[ip]:[port]/cam/realmonitor?[channel]&[subtype]

说明:

username: 用户名。例如 admin。

password: 密码。例如 admin。

ip: 为设备 IP。例如 10.7.8.122。

port: 端口号默认为 554，若为默认可不填写。

channel: 通道号，起始为 1。例如通道 2，则为 channel=2。

subtype: 码流类型，主码流为 0（即 subtype=0），辅码流为 1（即 subtype=1）。

例如，请求某设备的通道 2 的辅码流，Url 如下

rtsp://admin:admin@10.12.4.84:554/cam/realmonitor?channel=2&subtype=1

3、D-Link

rtsp://[username]:[password]@[ip]:[port]/[channel].sdp

说明：

username：用户名。例如 admin

password：密码。例如 12345，如果没有网络验证可直接写成 rtsp:// [ip]:[port]/[channel].sdp

ip：为设备 IP。例如 192.168.0.108。

port：端口号默认为 554，若为默认可不填写。

channel：通道号，起始为 1。例如通道 2，则为 live2。

例如，请求某设备的通道 2 的码流，URL 如下

rtsp://admin:12345@192.168.200.201:554/live2.sdp

4、Axis（安讯士）

rtsp://[username]:[password]@[ip]/axis-media/media.amp?[videocodec]&[resolution]

说明：

username：用户名。例如 admin

password：密码。例如 12345，如果没有网络验证可省略用户名密码部分以及@字符。

ip：为设备 IP。例如 192.168.0.108。

videocodec：支持 MPEG、h.264 等，可缺省。

resolution：分辨率，如 resolution=1920x1080，若采用默认分辨率，可缺省此参数。

例如，请求某设备 h264 编码的 1280x720 的码流，URL 如下：

rtsp:// 192.168.200.202/axis-media/media.amp?videocodec=h264&resolution=1280x720

5、海康威视

默认 IP 地址：192.168.1.64/DHCP 用户名 admin 密码自己设

端口："HTTP 端口"（默认为 80）、"RTSP 端口"（默认为 554）、"HTTPS 端口"（默认 443）和"服务端口"（默认 8000），ONVIF 端口 80。

RTSP 地址：rtsp://[username]:[password]@[ip]:[port]/[codec]/[channel]/[subtype]/av_stream

说明：

username: 用户名。例如 admin。

password: 密码。例如 12345。

ip: 为设备 IP。例如 192.0.0.64。

port: 端口号默认为 554，若为默认可不填写。

codec：有 h264、MPEG-4、mpeg4 这几种。

channel: 通道号，起始为 1。例如通道 1，则为 ch1。

subtype: 码流类型，主码流为 main，辅码流为 sub。

例如，请求海康摄像机通道 1 的主码流，Url 如下

主码流：

rtsp://admin:12345@192.0.0.64:554/h264/ch1/main/av_stream

子码流：

rtsp://admin:12345@192.0.0.64/mpeg4/ch1/sub/av_stream

6、大华

默认 IP 地址：192.168.1.108 用户名/密码：admin/admin

端口：TCP 端口 37777/UDP 端口 37778/http 端口 80/RTSP 端口号默认为 554/HTTPs 443/ONVIF 功能默认为关闭，端口 80

RTSP 地址：rtsp://username:password@ip:port/cam/realmonitor?channel=1&subtype=0

说明:

username: 用户名。例如 admin。

password: 密码。例如 admin。

ip: 为设备 IP。例如 10.7.8.122。

port: 端口号默认为 554，若为默认可不填写。

channel: 通道号，起始为 1。例如通道 2，则为 channel=2。

subtype: 码流类型，主码流为 0（即 subtype=0），辅码流为 1（即 subtype=1）。

例如，请求某设备的通道 2 的辅码流，Url 如下

rtsp://admin:admin@10.12.4.84:554/cam/realmonitor?channel=2&subtype=1

7、雄迈/巨峰

默认 IP 地址：192.168.1.10 用户名 admin 密码空

端口：TCP 端口：34567 和 HTTP 端口：80，onvif 端口是 8899

RTSP 地址：rtsp://10.6.3.57:554/user=admin&password=&channel=1&stream=0.sdp?

10.6.3.57 这个是被连接的设备的 IP

554 这个是 RTSP 服务的端口号，可以在设备的网络服务里面更改

user=admin 这个是设备的登录用户名

password= 密码空

channel=1 第一通道

stream=0.sdp？主码流

stream=1.sdp？副码流

图片抓拍地址：http://ip/webcapture.jpg?command=snap&channel=1

8、天视通

默认 IP 地址：192.168.0.123 用户名 admin 密码 123456

端口：http 端口 80 数据端口 8091 RTSP 端口 554 ONVIF 端口 80

RTSP 地址：主码流地址:rtsp://192.168.0.123:554/mpeg4

子码流地址:rtsp://192.168.0.123:554/mpeg4cif

需要入密码的地址： 主码流 rtsp://admin:123456@192.168.0.123:554/mpeg4

子码流 rtsp://admin:123456@192.168.0.123:554/mpeg4cif

图片抓拍地址：http://ip/snapshot.cgi

中维/尚维

默认 IP 地址：DHCP 默认用户名 admin 默认密码 空

RTSP 地址：rtsp://0.0.0.0:8554/live1.264（次码流）

rtsp://0.0.0.0:8554/live0.264 (主码流)

9、九安

RTSP 地址：rtsp://IP:port（website port）/ch0_0.264（主码流）

rtsp://IP:port（website port）/ch0_1.264（子码流）

10、技威/YOOSEE

默认 IP 地址：DHCP 用户名 admin 密码 123

RTSP 地址：主码流：rtsp://IPadr:554/onvif1

次码流：rtsp://IPadr:554/onvif2

onvif 端口是 5000

设备发现的端口是 3702

11、V380

默认 IP 地址：DHCP 用户名 admin 密码空/admin

onvif 端口 8899

RTSP 地址：主码流 rtsp://ip//live/ch00_1

子码流 rtsp://ip//live/ch00_0

12、宇视

默认 IP 地址： 192.168.0.13/DHCP 默认用户名 admin 和默认密码 123456

端口：HTTP 80/RTSP 554/HTTPS 110(443)/onvif 端口 80

RTSP 地址：rtsp://用户名:密码@ip:端口号/video123 123 对应 3 个码流

【注】zlmediakit 官方项目地址

4.4.3 消息引擎 emqx

消息引擎是基于 MQTT 协议的高性能实时通信枢纽，为设备与业务平台提供稳定可靠的双向通信能力。核心支撑以下业务场景：

4.4.3.1 核心应用场景

设备控制：毫秒级下发控制指令，用于智能楼宇的空调照明控制、工业设备启停、农业灌溉管理等，支持批量设备和指令状态追踪。

数据采集：实时采集设备状态、传感器读数、性能指标等数据，支持多频率采集和断点续传，用于监控和数据分析。

远程运维：实现设备远程诊断、故障预警、固件升级和配置更新，提升设备管理效率

4.4.3.2 emqx 连接配置

服务器地址: ip

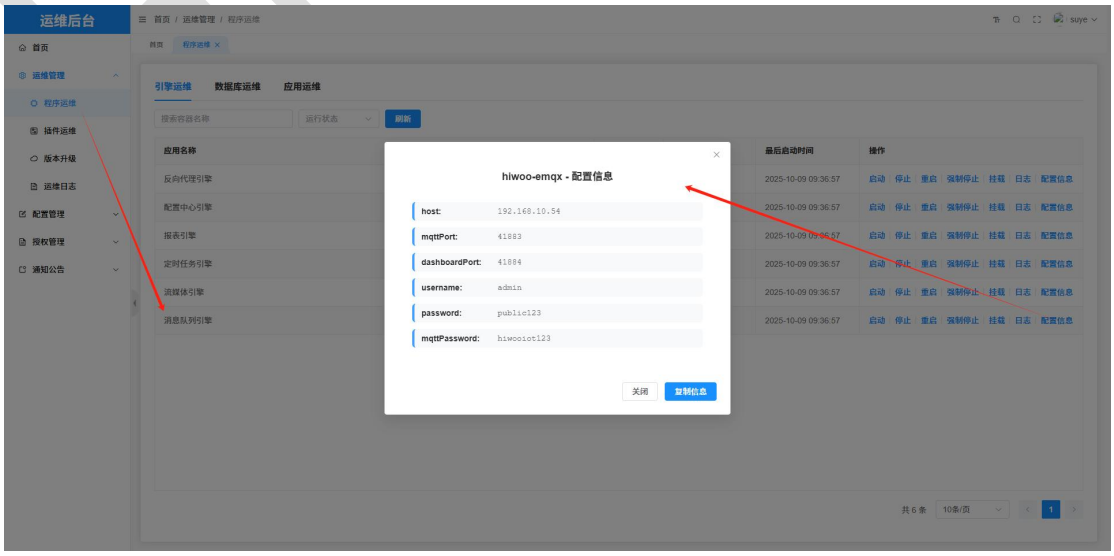
端口:port

协议: MQTT over TCP

用户名: admin

密码: Hiwooiot23

4.4.3.3 连接配置信息获取



点击消息队列引擎在配置信息按钮，即可弹出弹框显示配置连接信息。

4.4.3.4 多语言集成支持

消息引擎提供全面的多语言 SDK 支持，方便不同技术栈的开发者快速集成。

【注】EMQX 多语言 SDK 接入使用说明

4.5、OpenApi

OpenApi 是 Things 平台对外开放的标准 RESTful 接口集合，用于实现系统与第三方应用、设备或服务之间的数据交互与功能调用。通过 OpenApi，开发者可以灵活地集成设备管理、数据查询、指令下发等能力，满足多样化的业务集成需求。

4.5.1、接口认证

所有 OpenApi 接口均采用 Token 认证机制。调用前需先获取访问令牌（Access Token），并在请求头中携带该令牌进行身份校验。

4.5.1.1 获取访问令牌

调用本接口进行身份认证，成功后可获取访问**令牌 token**。该令牌是访问其他所有 API 的凭证。

描述	登录指定账号获取 token 与用户基本信息
接口地址	/api/hiwoo-base/uaa/user/login
请求方式	POST
请求头	Content-Type: application/json
请求参数	{"account": "您的用户名", "password": "您的密码","client": "pc/mobile"}
成功响应	{ "code": 200, "msg": "处理成功", "time": 1760443856385, "data": { "id": "1934509579304325122",

```
            "name": "suye999",
            "headSculpture": null,
            "account": "suye999",
            "roleId": "1",
            "roleName": null,
            "departmentId": "1934509578862297088",
            "departmentName": null,
            "enterpriseId": "HW6000002090647553",
            "phone": "19956132082",
            "email": null,
            "notes": null,
            "isLocked": 0,
            "isDeleted": 0,
            "password":
"RwRo0bBnQ7kUo+uG6UVhFL4E4NZcvL9/ym3qvtb66L7nqtI9x1ZFZH4pfzj62
frYFPSHYLS8aafdSxZPVAYVYidow1abWoQ50qUbjqJomPAKIKCHBGca+apvlxy
mpgSy+G3sCbOZ6tWqwOZZRvCCHfPVEjscFitPxqrRJWOUAZw=",
            "salt": null,
            "updateTime": "2025-10-14 20:10:56",
            "createTime": "2025-06-16 15:13:19",
            "tokenInfo": {
                "tokenName": "satoken",
                "tokenValue":
"66f05936-0c73-43a9-beab-d4ad69c16c1e",
                "isLogin": true,
                "loginId": "suye999",
```

<table>
<tr><td></td><td>
<pre>
"loginType": "login",
"tokenTimeout": 2592000,
"sessionTimeout": 2592000,
"tokenSessionTimeout": -2,
"tokenActiveTimeout": -1,
"loginDevice": "HW6000002090647553",
"tag": "HW6000002090647553"
},
"openid": "",
"miniProgramsOpenid": null,
"thirdPartyMiniProgramsOpenid": null,
"thirdPartyOpenid": null,
"companyName": "ad",
"superior": null
}
}
</pre>
</td></tr>
<tr><td>失败响应</td><td>
<pre>
{
  "code": 500,
  "msg": "登录失败，用户名或密码错误"
}
</pre>
</td></tr>
<tr><td>备注</td><td></td></tr>
</table>

4.5.1.2 在请求中携带令牌

获取到 token 后，您需要在调用**除登录接口外**的所有其他 OpenApi 接口时，将其设置在 HTTP 请求的 Header 中。

Header 名称： satoken

Header 值: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

示例：

描述	获取设备列表
接口地址	/api/hiwoo-iot/engine/device/get-device-list
请求方式	GET
请求头	Content-Type: application/json satoken: "eb2dc48e-feb1-4457-96e5-a99386273bbb"
请求参数	keyword=&status=&current=1&size=10&orgId=1869566543301783552&userId=1869566543750574082&tenantId=HW6000001415872484&roleId=1&order=desc
成功响应	{ "code": 200, "msg": "处理成功", "time": 1760616925638, "data": { "records": [{ "createTime": "2025-03-03 17:18:19", "updateTime": "2025-03-30 20:14:56", "state": 0, "deviceId": "1896490307579957248", "orgId": "1869566543301783552", "siteId": null,

```
        "deviceName": "1",
        "deviceSource": "Plus_Test",
        "deviceIllustrate": "",
        "status": "2",
        "isDisable": 0,
        "deviceType": "1",
        "deviceWay": "1",
        "reportingCycle": "5",
        "cycleUnit": "1",
        "address": null,
        "latitude": null,
        "longitude": null,
        "isSimulate": 1,
        "logoUrl": null,
        "creator": null,
        "tenantId": "HW6000001415872484",
        "userId": "dyp666",
        "isSyncAddress": 1,
        "deviceTag": null,
        "siteType": null,
        "deviceUniqueTag": null
    },
    {
        "createTime": "2025-07-04 23:29:48",
        "updateTime": "2025-07-04 23:29:48",
        "state": 0,
        "deviceId": "1941157503507021824",
        "orgId": "1869566543301783552",
        "siteId": null,
```

```
                "deviceName": "sdg",
                "deviceSource": "0",
                "deviceIllustrate": "",
                "status": "1",
                "isDisable": 0,
                "deviceType": "2",
                "deviceWay": "0",
                "reportingCycle": "1",
                "cycleUnit": "1",
                "address": "北京市东城区中华路甲 10 号",
                "latitude": 39.91473461,
                "longitude": 116.40397673,
                "isSimulate": 1,
                "logoUrl": "",
                "creator": null,
                "tenantId": "HW6000001415872484",
                "userId": "1869566543750574082",
                "isSyncAddress": 1,
                "deviceTag": null,
                "siteType": null,
                "deviceUniqueTag": null
            }
        ],
        "total": 2,
        "size": 10,
        "current": 1,
        "pages": 1
    }
}
```

失败响应	{
	"code": 500,
	"msg": "server error"
	}

备注	orgId:组织 id tenantId:租户 id roleId:角色 id

4.5.1.3 令牌过期与刷新

获取到的 token 具有有效期（见响应中的 tokenTimeout 字段，单位：秒）。在有效期内，该令牌可正常使用。

若令牌已过期，再次调用接口时会返回 401 状态码，提示 "Token 无效"。

本系统目前未提供自动刷新令牌的接口。当令牌过期后，客户端需要重新调用 **7.1.1 登录接口**以获取新的 token。

4.5.1.4 退出登录

描述	退出登录
接口地址	/api/hiwoo-base/uaa/user/logout
请求方式	POST
请求头	Content-Type: application/json
请求参数	{"account":"abcd","client":"PC","ip":"","address":""}
成功响应	{ "code": 200, "msg": "处理成功", "time": 1760443856385,

	"data": {} }

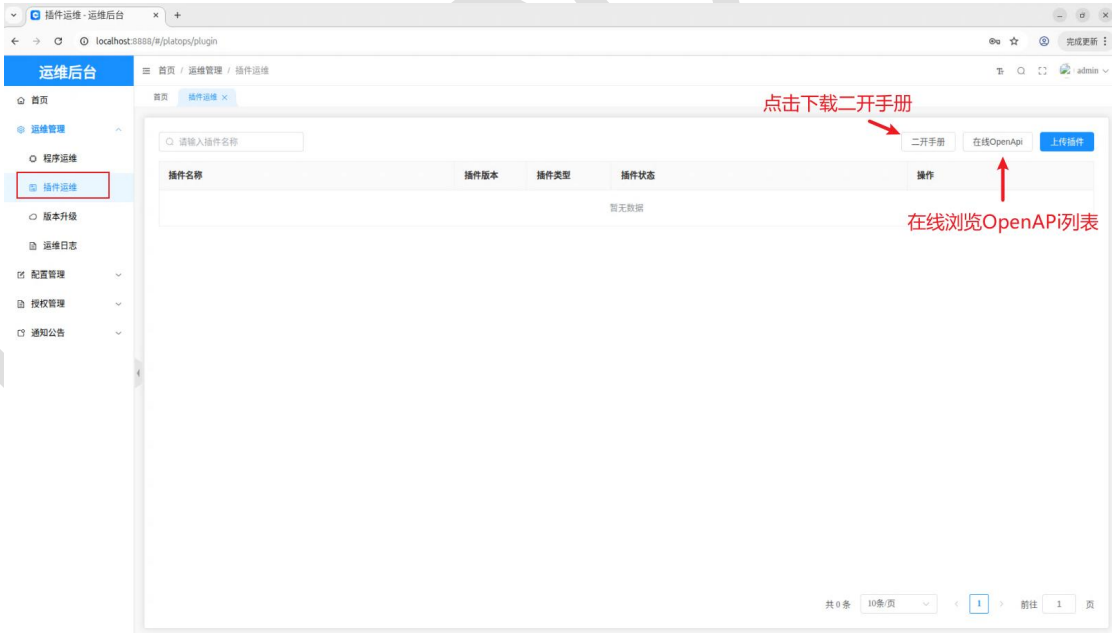
失败响应	{
	"code": 500,
	"msg": "server error"
	}

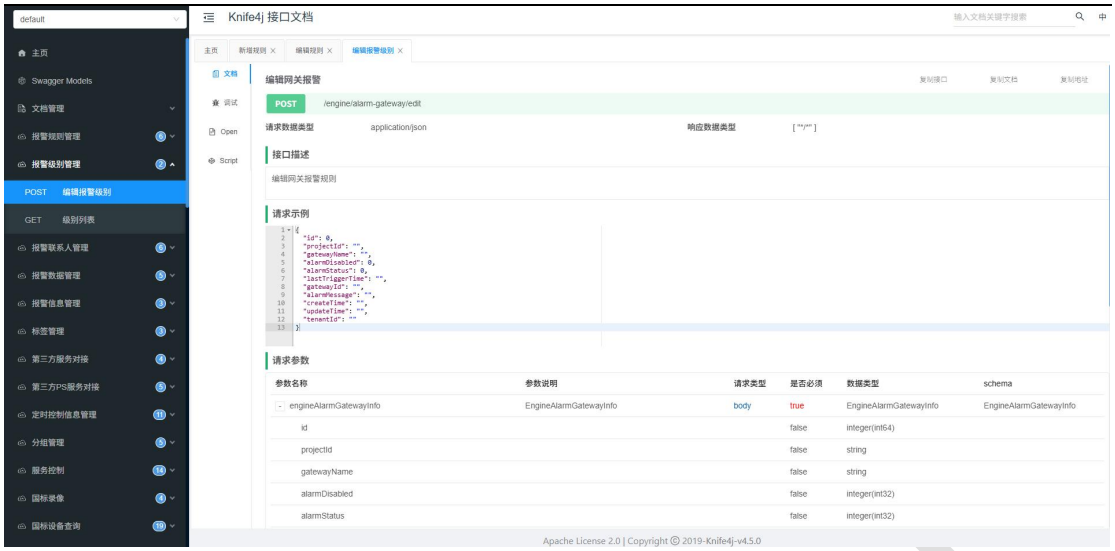
备注	在不再需要访问 API 时，可以调用此接口使当前令牌立即失效。

4.5.2、接口列表

4.5.2.1、在线接口（推荐）

详细接口列表请通过 **运维后台->插件管理->在线 OpenApi** 查看。

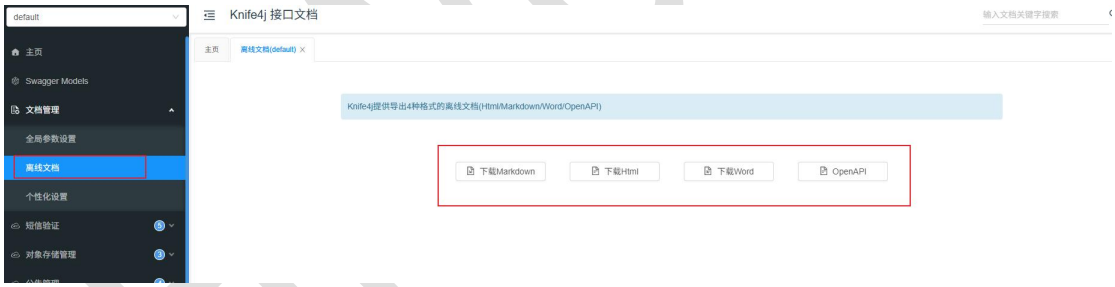




4.5.2.2、离线接口

物联数据中台.pdf 用户系统.pdf 数据可视化.pdf

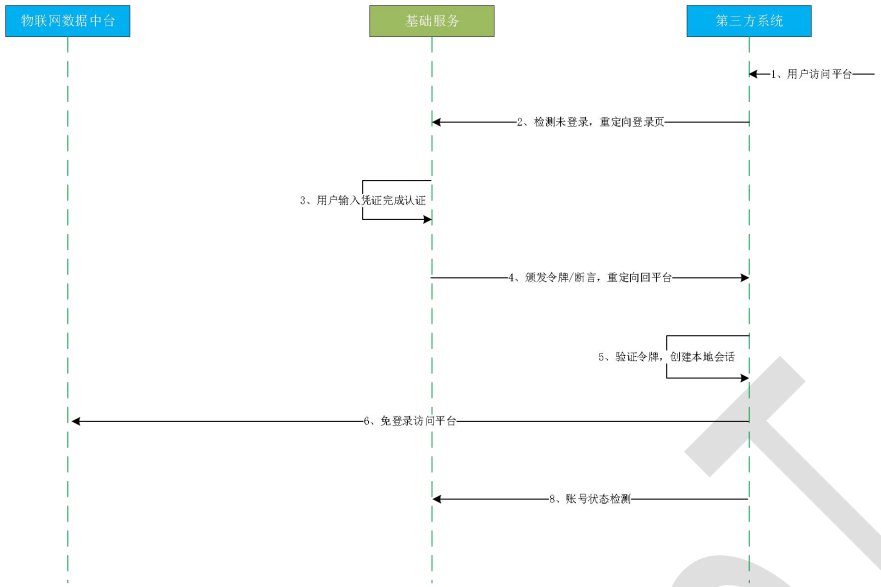
如需其他格式的接口文档，您也可以通过 4.5.2.1 在线接口文档自行下载



第五章 单点登录

5.1、应用场景

单点登录是一种身份认证方案，允许用户使用一套凭证（比如用户名和密码）登录多个独立的平台系统（包括子系统）。以 Things 物联网平台为中心实现一次登录，处处访问；一次登出，统一登出。



5.2、认证中心

平台统一登录认证中心。

认证中心地址	http://xxx/login;https://xxx/login
回调参数	redirect_uri:"https://xxx/xxx" client_id:"系统名称"
备注	认证中心地址：对应您所要对接的系统环境； redirect_uri：认证成功后所要回调的地址； client_id：系统识别，必须英文

5.3、Open API 接口

5.3.1、单点登出

使用方式	平台或子系统统一登出
接口地址	/api/hiwoo-base/uaa/v1/auth/logout
请求方法	POST
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求体	无

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760443856385,
	}

失败示例	{
	"code": 401,
	"time": 1760443856385,
	"msg": "您的凭证无效"
	}

备注	

5.3.2、状态检测

接口说明	平台或子系统周期检测状态保持
服务接口	/api/hiwoo-base/uaa/v1/auth/is-login
请求类型	GET
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求体	无

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760527748213,
	"data":true
	}

失败示例	{
	"code": 401,
	"time": 1760443856385,
	"msg": "您的凭证无效"
	}

备注	

5.3.2、获取用户信息

接口说明	根据凭证获取用户信息

服务接口	/api/hiwoo-base/uaa/v1/auth/user-info
请求类型	GET
请求类型	application/json

请求头	{
	"satoken": "您的访问凭证",
	...
	}

请求体	无

成功示例	{
	"code": 200,
	"msg": "处理成功",
	"time": 1760527748213,
	"data":{
	"id": "1939624641081868290",
	"name": "lang123",
	"headSculpture": null,
	"account": "lang123",
	"roleId": "1",
	"roleName": null,
	"departmentId": "1939624641084821504",
	"departmentName": null,
	"enterpriseId": "HW6000000127608400",
	"phone": "19856996057",
	"email": "hehaibo@hiwooiot.cn",
	"notes": null,
	"isLocked": 0,
	"isDeleted": 0,
	"companyName": "绿色环保解决方案",
	"openid": null,
	"miniProgramsOpenid": null,
	"thirdPartyMiniProgramsOpenid": null,
	"thirdPartyOpenid": null,
	}
	}

失败示例	{
	"code": 401,
	"time": 1760443856385,
	"msg": "您的凭证无效"
	}

备注	