HXGE 系列网关
LUA 脚本采集插件使用说明

北京宏达信诺科技有限公司

目录

目录 2

第 1 章 插件概述 2

第 2 章 插件使用详解 ..... **错误!未定义书签。**

第 3 章 脚本编辑与使用 ..... **错误!未定义书签。**

第 4 章 脚本应用示例 ..... **错误!未定义书签。**

第 1 章 插件概述

本插件基于 Lua 脚本语言，对物联网采集系统中的数据标签实现轻量化的边缘计算与边缘控制。插件可获取数据标签中 "double"、"string"、"float"、"long" 等数据类型的数值，并支持多标签之间的加、减、乘、除、比较、字符串拼接等多种运算，再将计算结果写入新定义的空标签中，或据此下发遥控指令，从而满足对采集数据进行相应计算以及对底层设备进行实时控制的功能需求。

变量说明

io.C1.D1.Tag1：其中 "io" 为连接标签前缀，默认添加 "io"；"C1" 为通道名称的英文名称；"D1" 为设备名称的英文名称；"Tag1" 为该设备下第一个标签的英文名称。按照由节点到根节点的格式依次连接，即为 io.C1.D1.Tag1。一般使用该字符串来识别本标签、获取标签内容。

第 2 章 插件使用详解

软件使用整体思路：打开软件，首先要在菜单栏找到工程管理，创建工程，在对应的工程目录下进行所有的操作，简要步骤为：创建工程->创建采集通道->创建设备->创建并配置采集标签点->创建转发通道并配置转发参数和标签->下载程序至网关->查看网关数据运行情况。下面详细图文操作步骤：

1、创建工程。

a).在菜单栏点击工程管理按钮，弹出工程管理悬浮框，新建工程组，如下图：



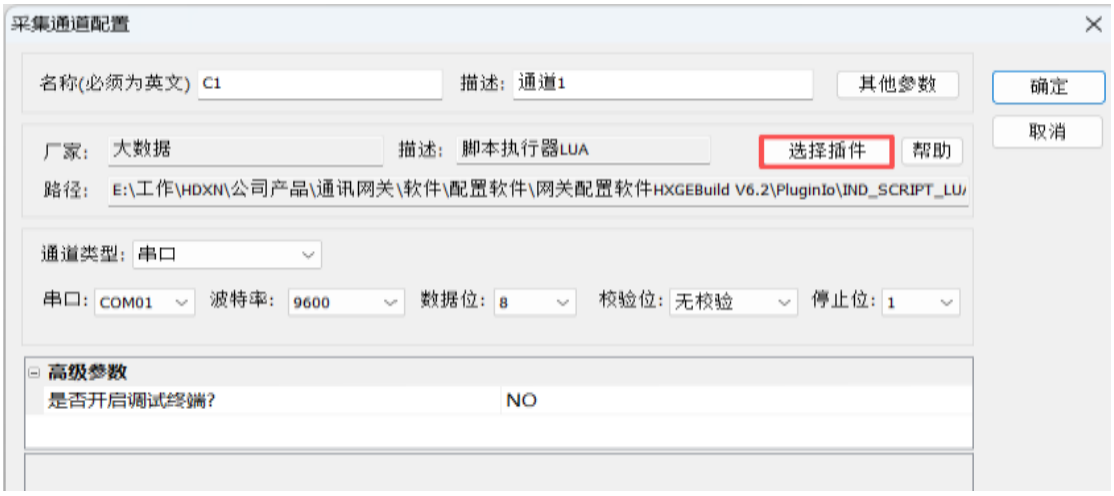
b).右击工程组名称新建工程，可自定义工程名称和描述，如下图：



2、创建采集通道。

a).在采集服务右击新建通道，选择插件，如下图：





b).在边缘计算目录下选择“脚本执行器LUA”插件，点击确定，如下图：



3、创建设备

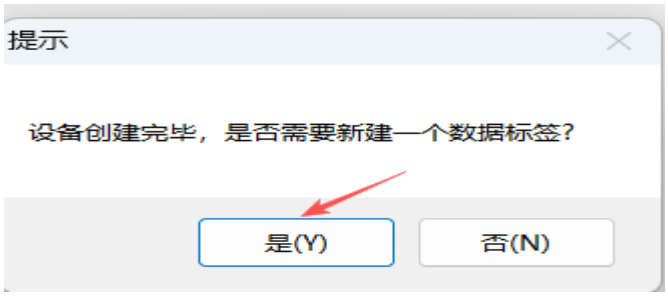
a).右击刚才新建的通道，新建设备，如下图：





注：脚本执行周期（毫秒）：指调用脚本的周期。

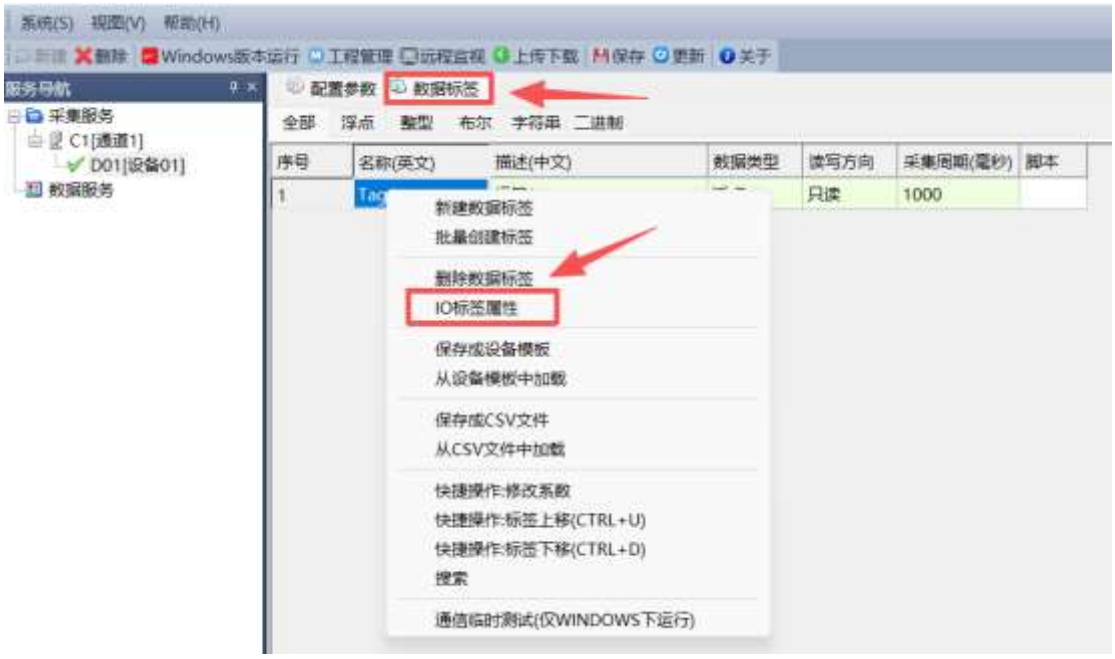
b）新建数据标签，点击“是”，如下图：



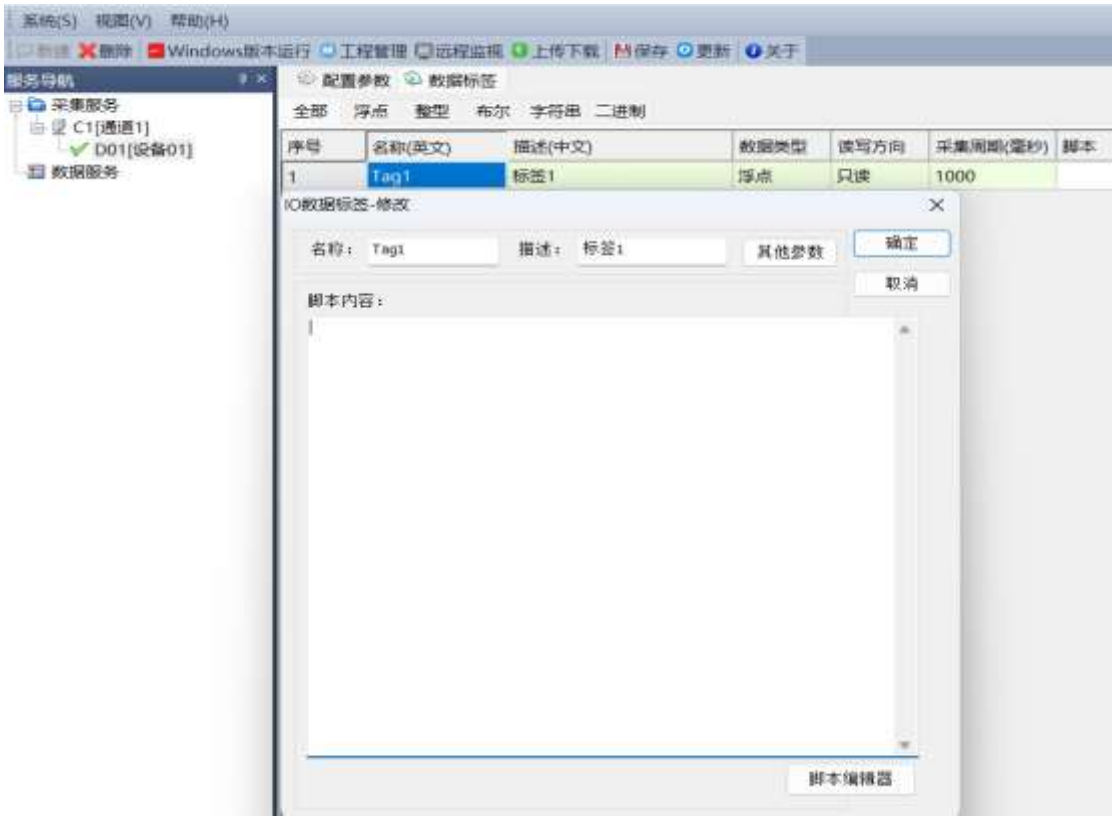
注：一个标签，执行一个javascript逻辑。

c）配置标签，如下图：

选中数据标签，右击标签，IO 属性打开标签。



在编辑框即可对脚本进行编辑，见下图：



第3章 脚本编辑与使用

脚本函数主要包含两类：第一类为 Lua 标准库提供的函数成员；第二类为自定义函数，包括用于与数据库进行交互的函数以及常用功能函数。以下介绍自定义函数：

- get_value_double(csTagLink)：获取单个点的double值,不含质量戳,只有1个链接项，1个值

  例：

  ```
  value = get_value_double ("io.C1.D1.Tag1");
  print(value);
  ```

- get_value_long(csTagLink)：获取单个点的long值,不含质量戳,只有1个链接项，1个值

  例：

  ```
  value = get_value_long ("io.C1.D1.Tag1");
  print(value);
  ```

- get_value_string(csTagLink)：获取单个点的string值,不含质量戳,只有1个链接项，1个值

  例：

  ```
  value = get_value_string ("io.C1.D1.Tag1");
  print(value);
  ```

- set_value _double(csTagLink, csValue)：仅对数据标签设定double值，对IO不进行写操作。

  例：

  ```
  set_value_double("io.C1.D1.Tag1","1234");
  ```

- set_value _long(csTagLink, csValue)：仅对数据标签设定long值，对IO不进行写操作。

  例：

  ```
  set_value_long("io.C1.D1.Tag1","1234");
  ```

- set_value _string(csTagLink, csValue)：仅对数据标签设定string值，对IO不进行写操作。

例：

```
set_value_string("io.C1.D1.Tag1","1234");
```

✧ yk_command (csTagLink, csValue)：对某个点进行遥控设定，该点读写属性需要为读写，或者只读。

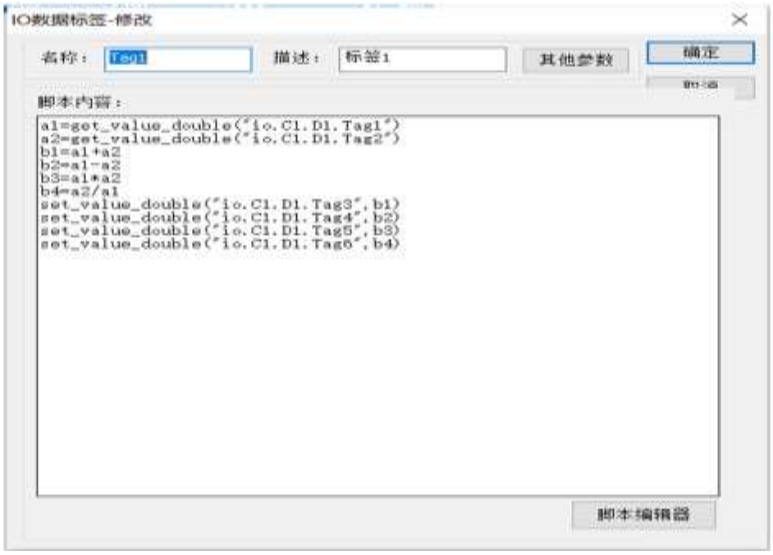
例：

```
yk_command ("io.C2.D1.Tag1","123");
```

第4章 脚本应用示例

利用仿真驱动创建六个点 io.C1.D1.Tag1、io.C1.D1.Tag2……io.C1.D1.Tag6，其中 Tag1 初始值设为 11，Tag2 初始值设为 22，用于模拟采集值。将 Tag1 与 Tag2 进行"+"运算，结果写入 Tag3；进行"-"运算，结果写入 Tag4；进行"*"运算，结果写入 Tag5；进行"/"运算，结果写入 Tag6。



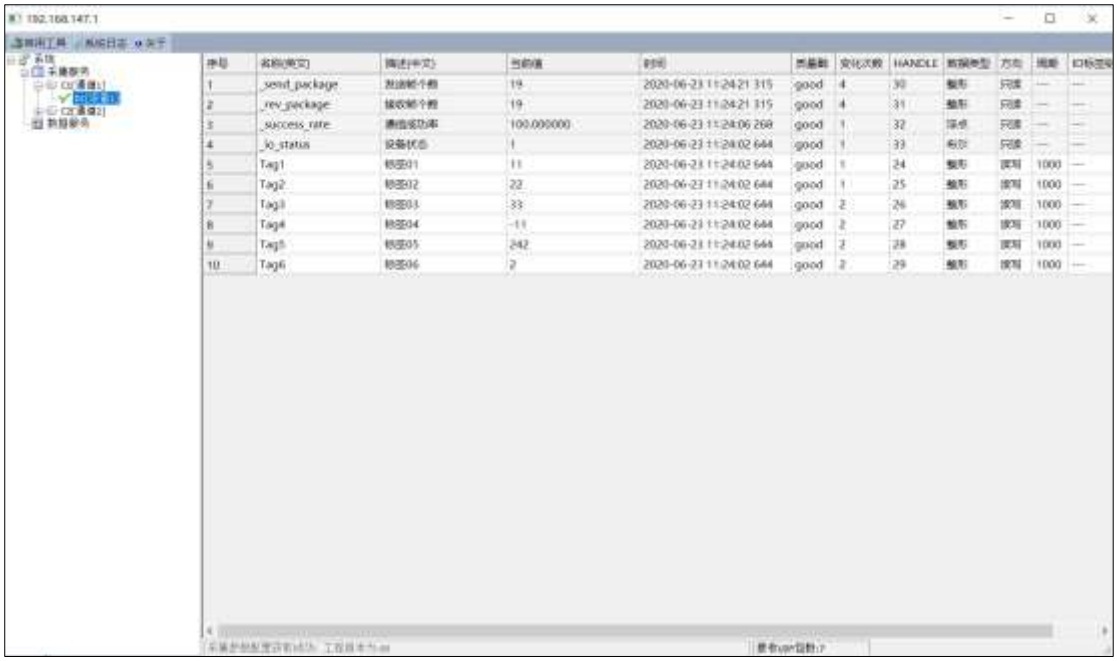


编辑脚本内容，脚本写完之后运行程序，查看脚本执行情况，查看脚本通道报文：执行成功，标识脚本语法正确。

```
a1=get_value_double("io.C1.D1.Tag1")
a2=get_value_double("io.C1.D1.Tag2")
b1=a1+a2
b2=a1-a2
b3=a1*a2
b4=a2/a1
set_value_double("io.C1.D1.Tag3",b1)
set_value_double("io.C1.D1.Tag4",b2)
set_value_double("io.C1.D1.Tag5",b3)
set_value_double("io.C1.D1.Tag6",b4)
```



查看通道一的 Tag3，Tag4，Tag5，Tag6 数值是否是脚本执行后的数据，经过计算，数据准确，lua 脚本编写完成。



附：Lua 脚本语言还提供更为丰富的计算函数与功能接口，此处不再逐一列举。实际使用中可根据具体业务需求，查阅并学习更多 Lua 脚本语言相关文档，自行编写符合需求的脚本内容。