

边缘计算平台WA-EDGE 插件与应用开发教程

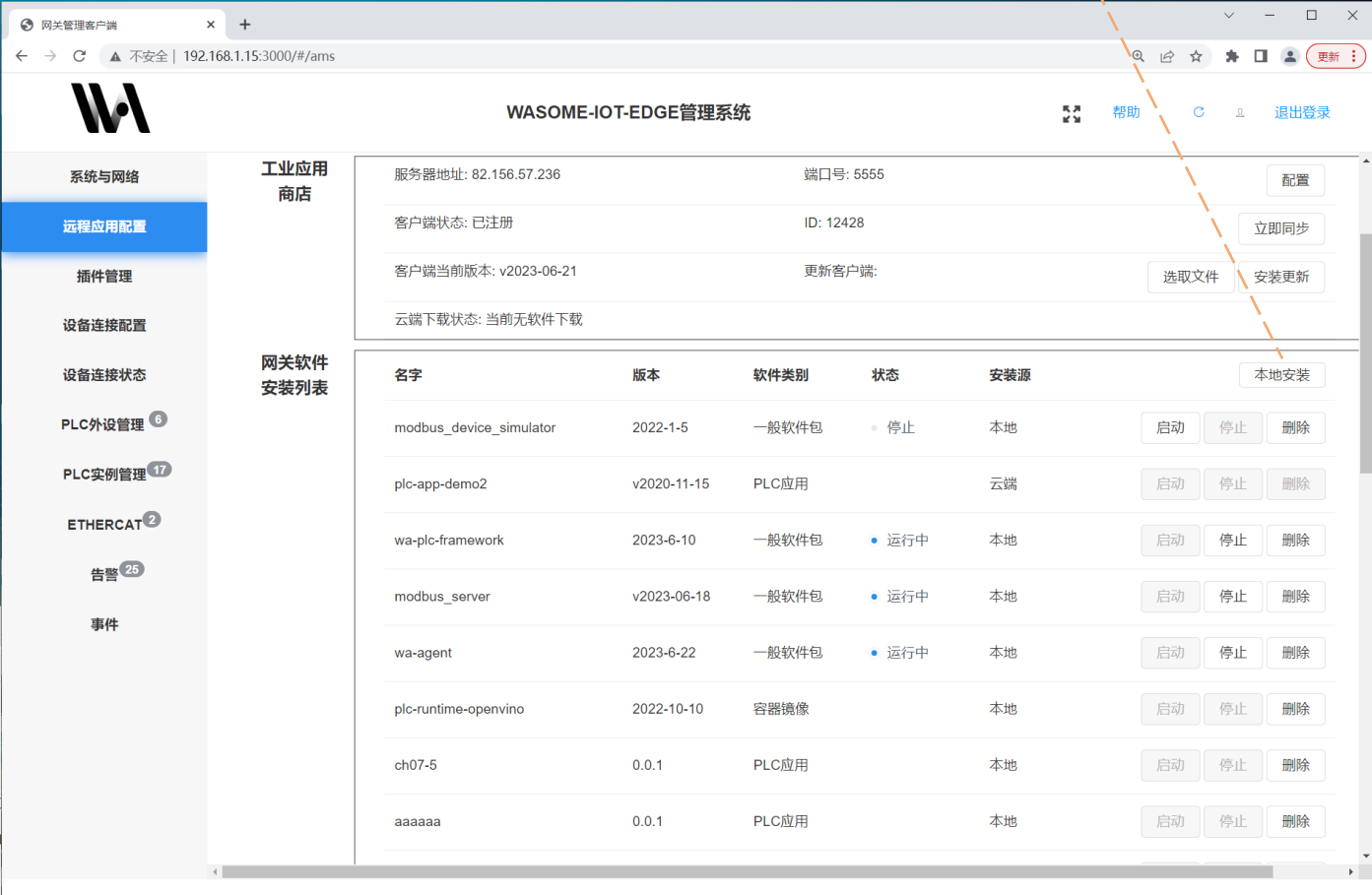
WA-EDGE开源社区平台 <https://gitee.com/wasome>

操作WA-EDGE边缘平台

用户从浏览器登录WA-EDGE的边缘管理系统 (WebConsole)，配置边缘平台的功能：

- 1. 动态安装、卸载用户开发边缘平台软件
- 2. 启动与停止软件
- 3. 插件的热插拔管理
- 4. 配置外部工业设备连接，查看设备连接状态，配置数据采集、及查看实时数据
- 5. PLC外设与实例管理
- 6. 告警与事件管理

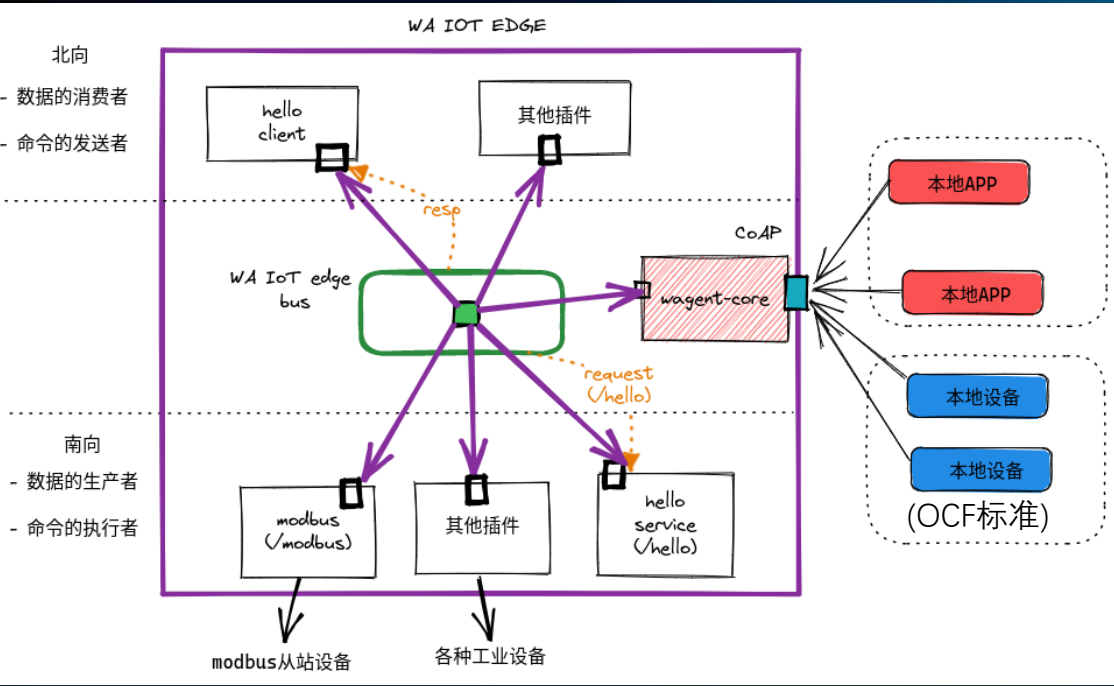
本地安装与管理边缘平台软件



使用插件配置虚拟OCF设备

WA-EDGE微服务插件框架

WA-EDGE插件框架

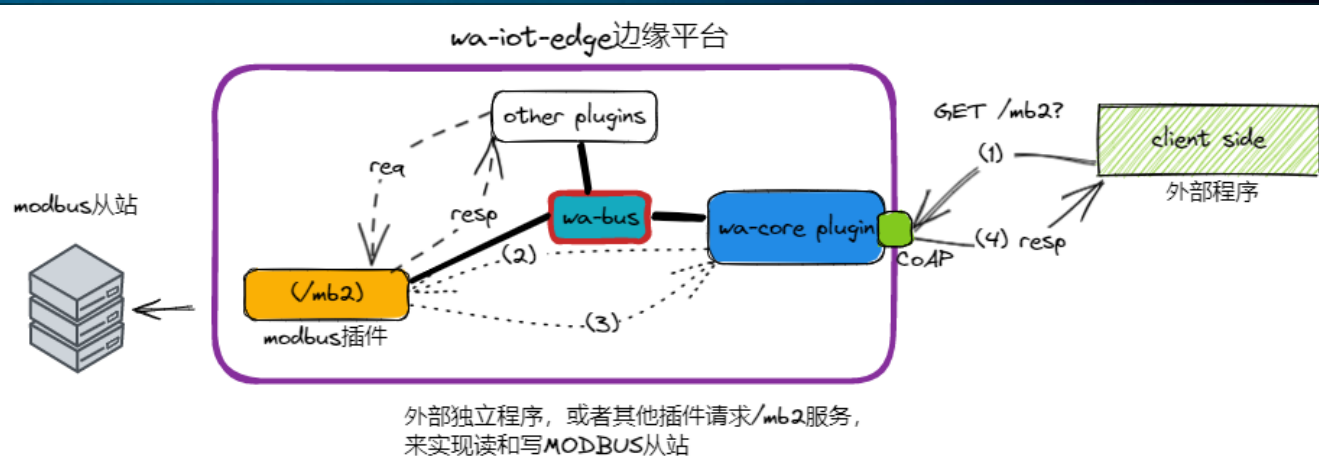


插件采用动态库文件格式，可以在框架中热插拔

提供基于微服务模式的插件开发框架。插件具有自包含特性，与使用场景解耦

用户可以为边缘平台开发插件、边缘应用(本地APP)，和采用OCF标准的IoT设备程序

插件微服务工作流



一个插件实现用URI (如上图/mb2)表示的若干资源, 提供微服务

微服务的请求和响应通过插件总线(WA-BUS)发送

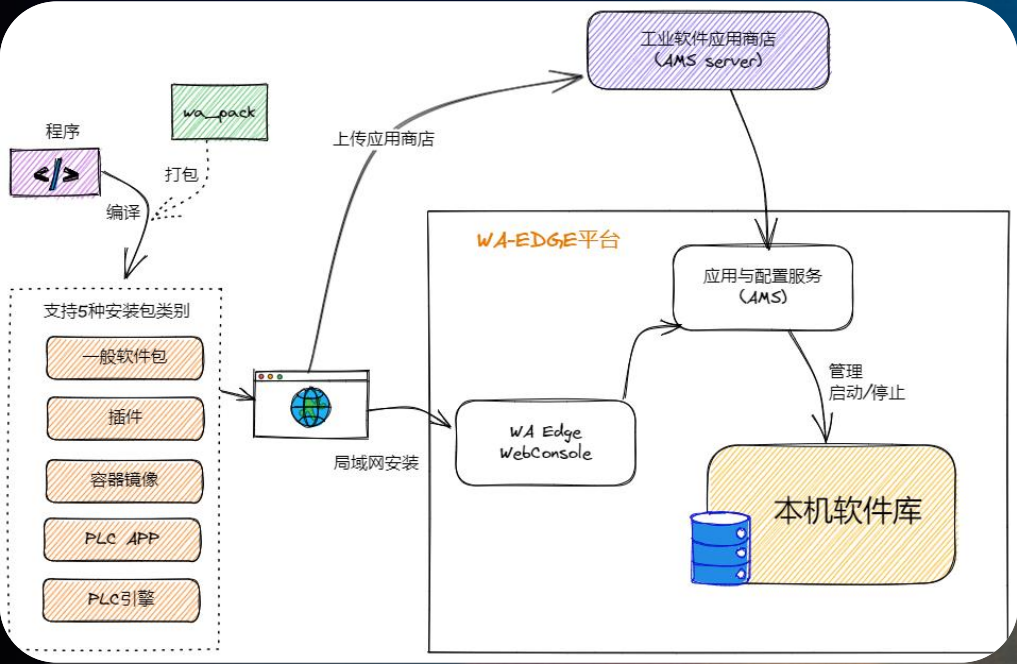
收到微服务请求后，插件可以对某物理设备进行读写操作，然后返回操作结果给请求者

插件请求之间可以相互请求微服务

系统插件“wa-core”使用UDP/CoAP端口对外提供微服务转发机制

WA-EDGE插件生态

WA-EDGE插件打包与分发



WA-EDGE平台提供多种格式的软件包管理，包含插件包
提供WA-PACK工具对开发的程序打包
支持云端的工业应用商店，远程管理设备的插件和其他软件

WA-EDGE插件管理系统

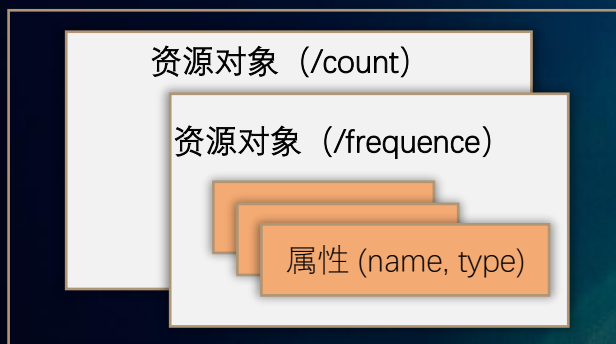


WA-EDGE的边缘平台管理系统，支持插件包的安装、卸载、升级
插件在WA-IOT-EDGE中热插拔

基于插件配置虚拟OCF设备

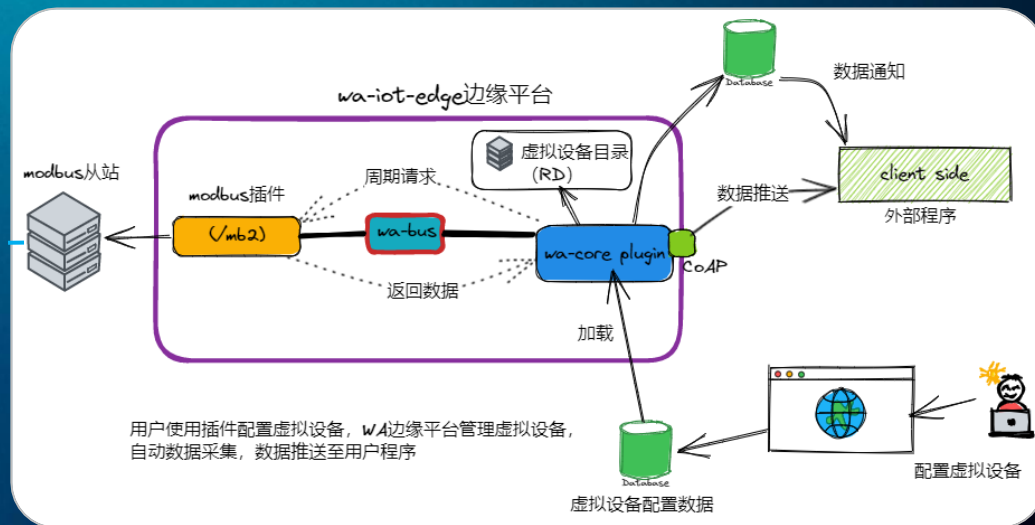
Open Connectivity Foundation (OCF) 是包含500多家全球会员的IoT标准开发组织

OCF定义设备与资源模型标准



将物理寄存器映射到OCF资源，实现异类设备统一管理

WA-EDGE平台使用OCF资源模型概念来表示所有物对象和资源目录(RD)管理物联网系统



WA-EDGE对RD中的物理和虚拟OCF设备提供操作配置、自动数采、以及数据流引擎等支持
提供SDK开发边缘程序

操作第1步：使用插件创建虚拟设备

第2步：配置OCF设备地址及资源

第3步：查看OCF设备和配置数采

添加虚拟设备

协议: modbus_plugin

uri: /mb2

设备名称: mb_dev1

设备类型: 请输入设备类型

取消 确定

设备名称: mb_dev1 协议(插件): modbus_plugin URI: /mb2

设备配置:

设备类型: RTU TCP 从站ID: 1

IP: 127.0.0.1 端口: 1502

资源(点位路径): 资源: /resource

插件清单(manifest)支持虚拟设备的个性化配置界面

点位名称	数据类型	条数	双字节编码	类型	地址
a	整型	请输入	请选择	Input	1
b	整型	请输入	请选择	Input	2

添加资源 清除

名称	状态	协议	进入状态时间	设备类型	应用配置
> d1502	on	modbus	2023-06-20 20:18:40	_dt_d1502_	☒ 本地
> mb2_1	on		2023-06-20 20:18:37		☒ 本地
▼ mb_dev1	on		2023-06-20 20:18:37		☒ 本地
资源: /resource 类型: 采集: ☒ 周期: 2					

插件开发：S7开源插件示例

源码：https://gitee.com/wasome/plugin_s7_plc

提示：更多的插件如modbus， AB PLC请参考社区开源仓库

s7插件提供资源“/s7”,实现对S7 PLC的读和写的访问

```
GET | PUT /s7?DI=[uuid]&ITEMS=[items]&area=[DB]&an=[area number]&addr=[address]&type=[data type]
```

插件回调框架

```
S7_PLC_APIS_all =  
{  
    {MODULE_API_VERSION_1},  
  
    S7_PLC_ParseConfigurationFromJson,  
    S7_PLC_FreeConfiguration,  
    S7_PLC_Create,  
    S7_PLC_Destroy,  
    S7_PLC_Receive,  
    S7_PLC_Start};
```

插件初始化

```
void S7_PLC_Start(MODULE_HANDLE module){  
  
    // 初始化微服务框架  
    RESTFUL_CONTEXT ctx = WA_InitRestful(  
        module_data->broker, module, "mod-ab-plc");  
    module_data->restful_context = ctx;  
  
    // 注册资源/s7以及处理函数入口  
    WA_RegisterResource(ctx, "/s7", res_s7_cmd_get_handler, T_Get);  
    WA_RegisterResource(ctx, "/s7", res_s7_cmd_put_handler, T_Put);  
  
    // 加载用户通过Web界面配置本插件的虚拟设备  
    load_device_config();  
  
    // 创建任务调度器(新线程). 创建S7连接任务  
    module_data->task_scheduler = bh_task_run_new_scheduler();  
    ...  
    bh_schedule_task(module_data->task_scheduler, task1, 0);  
}
```

资源实现函数

```
void res_s7_cmd_get_handler(restful_request_t *request,  
    REQ_ENV_HANDLE req_env){  
  
    // 解析请求的查询字段  
    char *di = parse_s7_cmd_query(request->query, &AREA,  
        &area_Number, &start_address, &items, &type);  
  
    // 查找目标设备名是否已经连接  
    device = find_device_open(di);  
  
    // 使用请求中的参数读S7设备  
    res = daveReadBytes(device->dc, AREA, area_Number, start_address,  
        length, NULL);  
  
    // 将读取结果返回请求者  
    wa_req_env_handover_payload(req_env, CONTENT_2_05,  
        IA_APPLICATION_JSON, strlen(payload), &payload);  
}
```

OCF设备程序开发

开发者除了使用插件创建OCF虚拟设备，还可以开发OCF设备程序

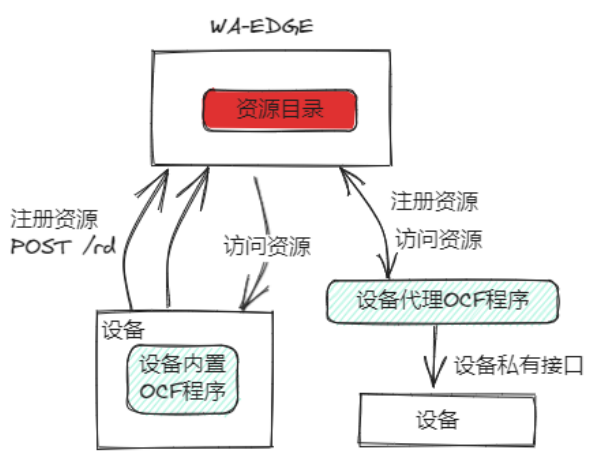
OCF程序可以设备内置，也可以是设备代理

OCF程序需实现两个主要场景：注册资源，处理资源请求

使用基于CoAP消息格式

WA-IOT-EDGE示例：endpoint（源码）

1. 设置当前设备的资源模型与资源实例
2. 向WA-EDGE的资源目录(RD)注册自身资源，维持定期更新，以保持联机状态
3. 实现资源的读写访问回调函数，处理外部对本设备资源的访问



Endpoint示例设备程序执行后，WA-EDGE平台收到其资源注册，对其可查看和配置

ble_reader1	离线	ep	2023-05-21 08:30:21	wasome.ble	本地	
资源: /count	类型: oic.kkk	采集: ☒	周期: 0			
资源: /frequency	类型: oic.abc	采集: ☒	周期: 3			

samples/clients/endpoint/main.cpp

```
// initiate the coap based restful framework
RESTFUL_CONTEXT restful_context = wa_coap_init_context(-1, 1000, 0);

// initiate the communication with wagent
wagent_ctx = wa_wagent_init(restful_context, "127.0.0.1");

// register two resources with the URI and handlers
wa_coap_register_resource(restful_context, RES_FRE, res_frequency, T_Default);
wa_coap_register_resource(restful_context, RES_CNT, res_count, T_Default);

// set resource type for the resources, so it can be used for registering on the wagent
wa_resource_set_type(restful_context, RES_FRE, "oic.abc");
wa_resource_set_type(restful_context, RES_CNT, "oic.kkk");

// prepare the resource directory report message
rd_assemble_t rd = rd_data_init_A(g_my_device_id, "ep", "wasome.ble", true);
// add all resource with type set using wa_resource_set_type()
wa_load_resources_to_rd(restful_context, rd);
// Tell the RD should updated every 10 seconds
rd_data_set_ttl(rd, 10);
// generate the message payload in JSON format
char * RD_message = rd_data_payload_A(rd);
rd_data_destroy(rd);

// register the RD on wagent
wa_wagent_set_rd(wagent_ctx, g_my_device_id, RD_message, 10*1000, rd_status_handler);
```

resource.cpp

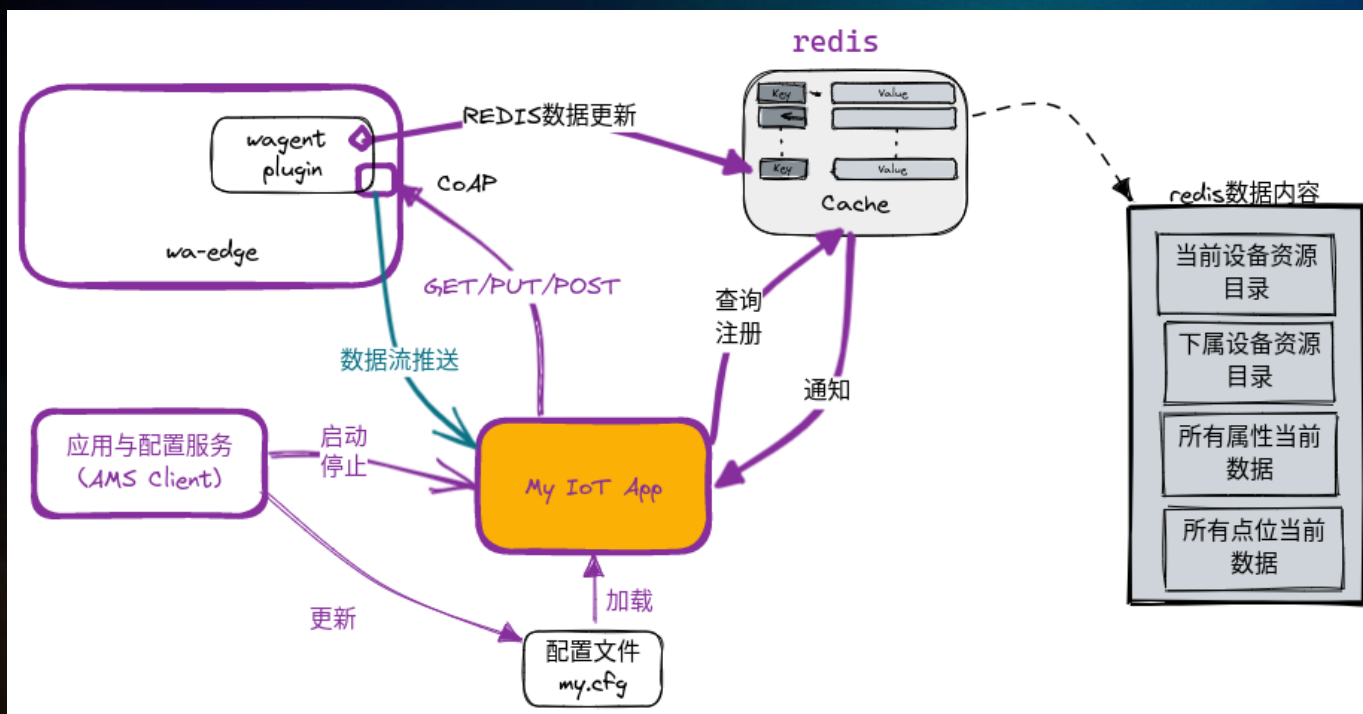
```
void res_frequency(restful_request_t * request, REQ_ENV_HANDLE req_env)
{
    char property[200] = {0};
    char buf[256];
    LOG2("Enter: method = %s", rest_action_str(request->action));
    if(request->action == T_Put && property[1])
    {
        g_frequency = atoi(request->payload);
        WALOG("frequency changed to %d", g_frequency);
        wa_set_response_status(req_env, CHANGED_2_04);
        return;
    }
}
```

WA-EDGE数据消费APP开发

- WA-EDGE提供操作界面，配置OCF资源自动数据采集 (如上页图)
- 最新数值会自动同步到REDIS
- 边缘应用获取数据的最简单方法是使用SDK中的REDIS API

WA-IOT-EDGE示例: electric-power (源码)

使用SDK开发ep-report程序，基于REDIS监听WA-EDGE读取的电表数据，然后基于电力104规约上报到云端



samples/clients/electric-power/main.cpp

```
// start monitoring the data
wa_redis_start_monitor(cb_redis_notify, (void*)NULL);
```

注册接收REDIS上的更新事件

```
void cb_redis_notify(REDIS_EVENT_T event, char * subject, void * content, void * user_data)
{
    WALOG("cb_redis_notify: sub: %s, content: %s\n", subject, content);

    if(event == Data_Value)
    {
        char * res = NULL;
        char * pt = NULL;
        char * di = wa_redis_subject_decompose(subject, &res, &pt);

        if(di == NULL || pt == NULL) return;
        read_from_redis(di, res, pt);
    }
    else if(event == AMS_Config_Update)
    {
        char * target_type = NULL;
        char * target_id = NULL;
        char * software_name = wa_redis_subject_AMS_CFG(subject, &target_type, &target_id);

        // the software name is what we defined product name in the package file "package.info"
        if(strcmp(software_name, "electric-power") == 0)
        {
            // cloud downloaded a new config file, reload it.
            load_configs();
        }
    }
}
```

REDIS上的数据更新触发

云端下发新的配置触发

调用SDK从REDIS读取目标资源的最新数据值

```
redis_property_value_t* values = wa_redis_read_resource(c, di, res, &size);
if(values)
{
    WALOG("read electric power data: di=%s, res=%s", di, res);
    report_ep104_data(values, size);
    wa_redis_free_values(values, size);
}
```

WA-EDGE 数据流引擎应用开发

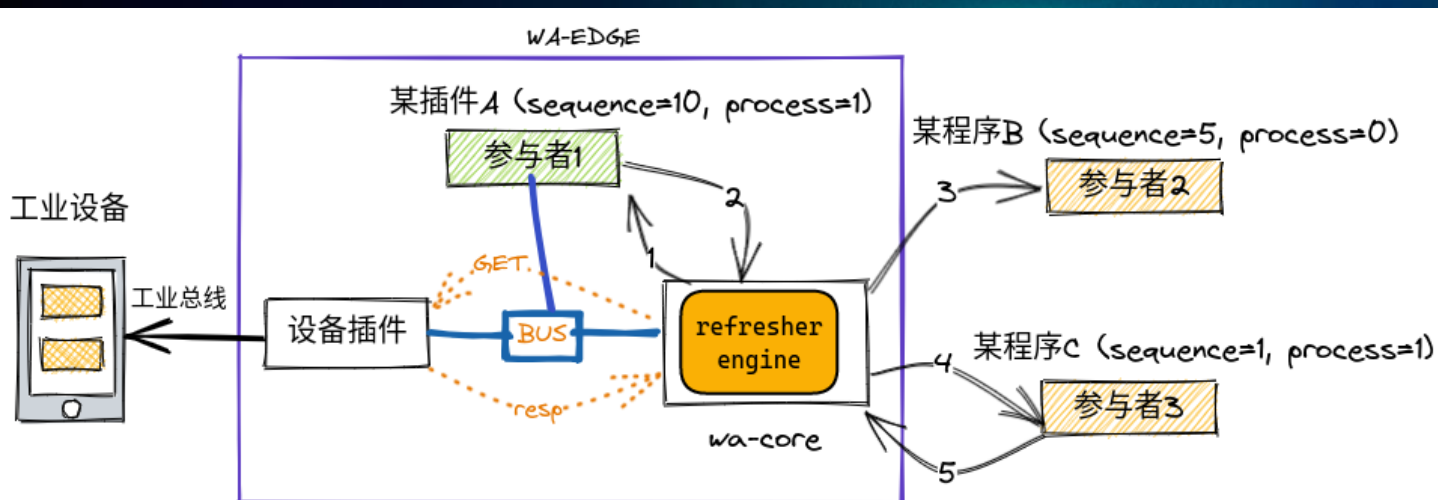
解耦共同参与数据处理流程的多方应用

每个参与者可以决定一个数据的命运

- 继续下一个流程节点处理
- 修改数据内容交给下一个节点处理
- 终止当前数据处理流程

WA-IOT-EDGE示例：data-flow (源码)

展示基于CoAP接口监听WA-EDGE所有设备数据，并进行数据流的处理



- 数据流参与者可以是某插件，或者WA-EDGE的外部程序
- 数据采集引擎周期采集数据，根据数据流的参与者的sequence数值大小依次发布数据，
- 如果参与者的参数process=0，不等待其回复继续下一位。
- 参与者回复中可以决定本数据命运：1) 继续 2) 扔弃本数据 3) 修改后继续

Data-flow示例源码

```
// initiate the coap based restful framework
RESTFUL_CONTEXT restful_context = wa_coap_init_context(-1, 1000, 0);

// initiate the communication with wagent
wagent_ctx = wa_wagent_init(restful_context, "127.0.0.1");

WA_DATA_MONITOR global_mon = wa_monitor_global_new(wagent_ctx, my_app_id, "tag1", data_global_handler);
wa_monitor_set_process(global_mon, true);
status = wa_monitor_run(global_mon);
WARNING2("Start global monitor %s", status?"OK":"FAIL");

// monitor the resource from sample "endpoint"
WA_DATA_MONITOR mon1 = wa_monitor_res_new(wagent_ctx, my_app_id, "tag2", "ep_001", "/frequency", data_mon1_handler);
wa_monitor_set_interval_ms(mon1, 2000);
wa_monitor_set_process(mon1, true);
wa_monitor_set_sequence(mon1, 10);
status = wa_monitor_run(mon1);
WARNING2("Start monitor #1: %s", status?"OK":"FAIL");

// start the main loop for the restful framework handling all the messages
while(1)
{
    // check the expiry of transactions
    uint32_t near_task_time = wa_coap_check_expiry(restful_context);
    if(near_task_time == -1) near_task_time = 1000;

    // doing blocking wait and ensured wakeup by the nearest expiration
    wa_coap_recieve_process(restful_context, (int)near_task_time);
};

void data_mon1_handler(restful_request_t * request, REQ_ENV_HANDLE req_env)
{
    wa_data_notify_t * notify = wa_monitor_parse_notification_A(request);
    if(notify == NULL)
    {
        WARNING2("Invalid data notify. query: %s", request->query?request->query:"null");
        return;
    }

    if(value > 10)
    {
        char buf[100];
        snprintf(buf, sizeof(buf), "%d", 10);
        int code = wa_monitor_get_decision_code(WA_Process_Modified);
        wa_set_response(req_env, code, TEXT_PLAIN, strlen(buf), buf);
        return;
    }
    else if(value < 1)
    {
        wa_monitor_set_decision(req_env, WA_Process_Drop);
    }
    else
    {
        wa_monitor_set_decision(req_env, WA_Process_Continue);
    }
}
```

注册接收某设备的数据，参与数据流处理

WA-EDGE上有数据时，回调函数自动触发

让WA-EDGE使用修改的数据进入下一个处理

让WA-EDGE扔弃当前数据

让WA-EDGE继续当前数据的下一个处理

<https://gitee.com/wasome>