IoT 边缘

SDK 参考

文档版本 01

发布日期 2024-10-21

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 SDK 下载

资源包名	描述	语言版本	版本	下载路径
Module SDK	Module SDK是开发边缘运行应用（插件）所必须的工具包，提供数据处理、协议转换、IT子系统接入等功能，开发完成后，通过选择打包方式来决定是容器化部署还是进程化部署。	Java版	2.2.7.Release	ModuleSDK_Java
		C版	/	ModuleSDK_C_latest(包括x86_64, arm32, arm64版本,下载后解压选择对应版本)
		C#版	1.0.7	ModuleSDK_CSharp

2 JAVA 版 SDK

SDK获取和安装

1. 安装Java开发环境。

访问**Java官网**，下载并说明安装Java开发环境。

说明

华为云Java SDK支持Java JDK 1.8 及其以上版本。

2. 使用eclipse/IDEA创建工程。

3. 下载ModuleSDK，并在工程中导入jar包。

4. 开发代码

开发数据处理的代码示例，详细说明请参考**开发应用集成ModuleSDK进行数据处理**。

```
/**
 * 监控APP，检视设备上报的数据，并对设备进行相应的控制
 */
public class MonitorApp implements BusMessageCallback {
    /**
     * 接收设备数据的消息总线输入点，取值需在创建应用版本的inputs参数中定义
     */
    private static final String INPUT = "input";
    /**
     * 发送设备数据的消息总线输出点，取值需在创建应用版本的outputs参数中定义
     */
    public static final String OUTPUT = "output";
    public static final int FIVE_SECOND = 5000;
    /**
     * 电机设备的产品ID
     */
    public static final String MOTOR_PRODUCT_ID = "6b4843db3f0189e9c577";

    /**
     * 与EdgeHub通信的客户端
     */
    private AppClient appClient;

    public MonitorApp() throws GeneraException {
        appClient = AppClient.createFromEnv();
    }

    public void start() throws GeneraException {
        //设置回调，打开客户端
        appClient.setBusMessageCallback(INPUT, this);//设置收到设备数据的回调
```



```
        appClient.open();
    }

    public void stop() throws GeneraException {
        appClient.close();
    }

    /**
     * 收到设备上报数据的回调处理，样例代码在马达设备状态错误时对马达进行重启
     *
     * @param busMessage
     */
    @Override
    public void onMessageReceived(BusMessage busMessage) {
        try {
            if (busMessage.getProductId().equals(MOTOR_PRODUCT_ID)) {
                //马达设备状态错误时对马达进行重启
                MotorData motorData = JsonUtil.fromJson(
                    JsonUtil.toJson(busMessage.getServices().get(0).getProperties()), MotorData.class);
                if (motorData.getStatus().equals("error")) {
                    Command command = new Command(busMessage.getDeviceId(), "power", "restart", null);
                    appClient.callDeviceCommand(command, FIVE_SECOND);

                }else{
                    //其他设备数据发布到总线
                    appClient.sendBusMessage(OUTPUT, busMessage);
                }
            } else {
                //其他设备数据发布到总线
                appClient.sendBusMessage(OUTPUT, busMessage);
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

3 C 版

SDK获取和安装

1. 安装cmake开发环境。

通过命令安装，也可以手动下载，下载地址：

https://cmake.org/download/

说明

华为云C SDK支持cmake 3.9.5 及其以上版本。

2. 使用CLion创建工程。

3. 下载ModuleSDK。

4. 开发代码

开发数据处理的代码示例，详细说明请参考**开发应用集成ModuleSDK-C进行数据处理**。

Demo实现的流程如下：

步骤1 通过edge_init初始化工作目录。

步骤2 通过edge_set_callbacks设置回调函数。

步骤3 Demo中只使用到on_message_received_cb回调函数，只需修改on_message_received_cb即可。

步骤4 通过edge_login初始化SDK，包括连接环境变量，连接Hub，订阅Topic，设置回调。

步骤5 通过set_bus_message_cb调用edge_set_bus_message_cb，SDK会根据input_name订阅Topic(比如/modules/user_monitor_app/messages/inputs/input，这里user_monitor_app是SDK应用对应的模块id，最后的“input“就是Demo代码里的input_name)，这个函数会将on_message_received_cb作为回调函数。

步骤6 回调函数on_message_received_cb里调用edge_send_bus_message，将未处理的数据发送回消息总线，设置该函数里的output_name，边缘Hub会订阅类似/modules/user_monitor_app/messages/outputs/output的Topic（这里user_monitor_app是SDK应用对应的模块id，最后的“output“就是Demo代码里的output_name）。

步骤7 调用设备命令，只有当设置的MOTOR_PRODUCT_ID的当前上报数据的设备的产品ID吻合，并且显示状态为error时，通过edge_call_device_command调用设备命令将设备重启。

步骤8 处理过程结束。

----结束

```
#include "edge.h"

#include <stdio.h>
#include <string.h>
#include <unistd.h>

/*
 * 描述：设置总线消息回调，用于对设备上报的数据进行处理
 * 参数：
 *  input_name：消息总线输入点
 * */
EDGE_RETCODE set_bus_message_cb(const char* input_name)
{
    edge_set_bus_message_cb(input_name);
    printf("set bus message callback with input name: %s\n", input_name);

    return EDGE_SUCCESS;
}

/*
 * 描述：收到设备上报数据的回调处理，样例代码在马达设备状态错误时对马达进行重启
 * 参数：
 *  device_id：设备ID
 *  product_id: 产品ID
 *  body: 上报的数据
 *  body_len: 上报数据的大小
 * */
EDGE_RETCODE on_message_received_cb(const char* device_id, const char* product_id, const char* body,
unsigned int body_len)
{
    // 设置发送设备数据的消息总线输出点，取值需在创建应用版本的outputs参数中定义
    char* output_name = "output";

    printf("start send message to output topic: %s\n", output_name);
    printf("body is: %s\nbody len is: %d\n", body, body_len);
    printf("product_id is: %s\n", product_id);
    printf("start processing device.\n");

    // 设置电机设备产品ID
    char* MOTOR_PRODUCT_ID = "product_123";

    if (strcmp(product_id, MOTOR_PRODUCT_ID) == 0)
    {
        //马达设备状态错误时对马达进行重启
        char* error = "error";
        char* is_error = strstr(body, error);

        // 设置默认超时时间
        unsigned int timeout = 5;
        ST_COMMAND command = {0};
        command.object_device_id = device_id;
        command.service_id = "power";
        command.command_name = "restart";

        // 调用设备命令重启
        if (is_error != NULL) edge_call_device_command(&command, timeout);
    }
    else {
        //其他设备数据发送到消息总线
        edge_send_bus_message(output_name, body, body_len);
    }
```



```
    printf("process ended.\n");

    return EDGE_SUCCESS;
}

/*
 * 监控APP，检视设备上报的数据，并对设备进行相应的控制
 */

void monitor_app()
{
    // 禁用缓冲区
    setvbuf(stdout, NULL, _IONBF, 0);

    printf("start monitor app\n");

    //初始化sdk，工作路径设置（工作路径下需要含有 /conf 目录（该目录下包含证书等信息））
    edge_init("../code/api_test/workdir");

    ST_MODULE_CBS module_cbs = {0};
    ST_DEVICE_CBS device_cbs = {0};

    module_cbs.pfn_on_message_received_cb = on_message_received_cb;

    // 设置回调函数
    edge_set_callbacks(&module_cbs, &device_cbs);

    printf("SDK start running!\n");

    sleep(1);
    edge_login();

    sleep(1);

    // 接收设备数据的消息总线输入点，取值需在创建应用版本的inputs参数中定义
    char* input_name = "input";
    set_bus_message_cb(input_name);

    // 这里是为了使应用能够长时间运行
    while(1)
    {
        sleep(1000);
    }

    edge_logout();
    sleep(1000);
    edge_destroy();
}


int main()
{
    // 监控app demo
    monitor_app();

    return 0;
}
```

