事件网格

SDK 参考

文档版本 01

发布日期 2024-02-23

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。

华为云计算技术有限公司

地址： 贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址： https://www.huaweicloud.com/

目 录

1 SDK 概述

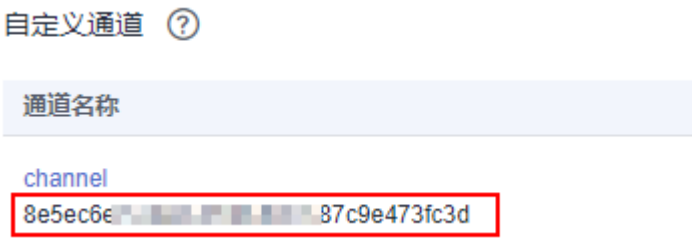
本文主要介绍事件网格提供的开源CloudEvents SDK，以简化您的开发工作。

2 收集信息

在使用开源的CloudEvents SDK发布事件前，您需要先获取以下信息。

通道 ID

在事件网格控制台的左侧导航栏中选择“事件通道”，通道名称下的一串字符即为通道ID，如下图所示。

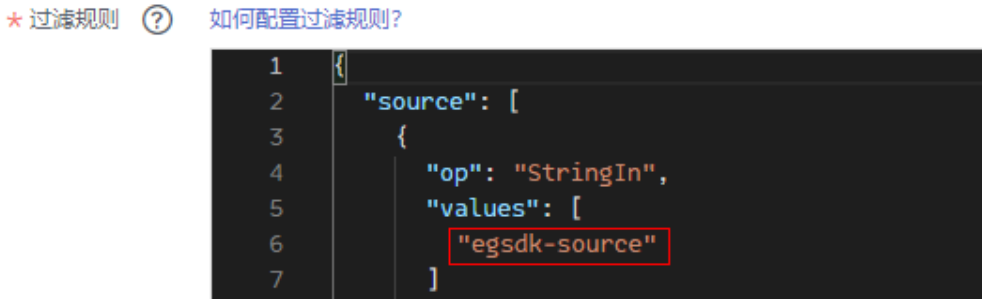


过滤规则中的 values 值

步骤1 在事件网格控制台的左侧导航栏中选择“事件订阅”，单击订阅名称，进入订阅详情页。

步骤2 单击事件源，弹出“事件源”对话框。

步骤3 在“过滤规则”中查看“values”值，如下图所示。



----结束

3 CloudEvents SDK

本章节介绍使用开源的CloudEvents Java SDK发布事件。

前提条件

1. 获取并安装IntelliJ IDEA，如果未安装，请至**IntelliJ IDEA官方网站**下载。
2. 在**pom.xml**中加入依赖。如何在Java环境中集成API请求签名的SDK请参考**AK/SK签名认证操作指导**。

```
<dependency>
        <groupId>io.cloudevents</groupId>
        <artifactId>cloudevents-json-jackson</artifactId>
        <version>${cloudevents.version}</version>
</dependency>
<dependency>
        <groupId>com.huawei.apigateway</groupId>
        <artifactId>java-sdk-core</artifactId>
        <version>3.1.2</version>
</dependency>
```

说明

“cloudevents.version”使用最新版本2.2.0。

发布事件

发布事件的示例代码如下（以下加粗内容需要请根据实际情况替换）：

```
import com.alibaba.fastjson.JSONObject;
import io.cloudevents.CloudEvent;
import io.cloudevents.core.builder.CloudEventBuilder;
import io.cloudevents.core.provider.EventFormatProvider;
import io.cloudevents.jackson.JsonFormat;
import lombok.extern.slf4j.Slf4j;
import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ssl.NoopHostnameVerifier;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContexts;
import org.apache.http.util.EntityUtils;
```

```
import java.net.URI;
import java.time.LocalDateTime;
import java.time.OffsetDateTime;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.stream.Collectors;

@Slf4j
public class PublishService {
    private static final String NAME = "l******";

    private static final String PASSWORD = "******";

    private static final String DOMAIN_NAME = "paa******01";

    private static final String PROJECT_ID = "c5******f";

    private static final String CHANNEL_ID = "ff******3";

    private static final String ENDPOINT = "events.ap-southeast-1.myhuaweicloud.com";

    private static final String URL = "https://" + ENDPOINT + "/v1/" + PROJECT_ID + "/channels/" +
CHANNEL_ID + "/events";

    private static final String SOURCE = "test_source";

    private static final String IAM_ENDPOINT = "iam.ap-southeast-1.myhuaweicloud.com";

    private static final String IAM_ADDRESS = "https://" + IAM_ENDPOINT + "/v3/auth/tokens";

    private static final String IAM_BODY = "{\"auth\": {\"identity\": {\"methods\": [\"password\"],\"password
\": {\"user\": " +
            "{\"name\": \"" + NAME + "\",\"password\": \"" + PASSWORD + "\",\"domain\": {\"name\": \"" +
DOMAIN_NAME + "\"}}}}," +
            "\"scope\": {\"project\": {\"id\": \"" + PROJECT_ID + "\"}}}}";

    private static final Map<String, String> tokenCache = new HashMap<>();

    private static final Map<String, LocalDateTime> expireTimeCache = new HashMap<>();

    public static void main(String[] args) {
        try {
            PublishService publishService = new PublishService();
            CloudEvent cloudEvent = publishService.buildCloudEvent(SOURCE);
            publishService.publish(cloudEvent);
        } catch (Exception exception) {
            exception.printStackTrace();
        }
    }

    /**
     * 获取token
     *
     * @param userName IAM账户用户名
     * @param domainName IAM账户domain用户名
     * @throws Exception
     */
    public String getToken(String userName, String domainName) throws Exception {
        synchronized(expireTimeCache) {
            String cacheKey = domainName + userName;
            LocalDateTime expireTime = expireTimeCache.get(cacheKey);
            if (expireTime != null && tokenCache.get(cacheKey) != null) {
                LocalDateTime curTime = LocalDateTime.now().minusMinutes(5);
                if (curTime.isBefore(expireTime)) {
                    return tokenCache.get(cacheKey);
                }
            }
            Map<String, String> headers = new HashMap<>();
            HttpResponse postResponse = getPostResponse(IAM_ADDRESS, headers, IAM_BODY);
```



```
            Header[] allHeaders = postResponse.getAllHeaders();
            HttpEntity entity = postResponse.getEntity();
            if (entity != null) {
                String entityString = EntityUtils.toString(entity, "UTF-8");
                JSONObject jsonObject = JSONObject.parseObject(entityString);
                DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("yyyy-MM-
dd'T'HH:mm:ss");
                String[] split = jsonObject.getJSONObject("token").getString("expires_at").split("\\.");
                expireTimeCache.put(cacheKey, LocalDateTime.parse(split[0], dateTimeFormatter));
            }

            List<Header> collect = Arrays.stream(allHeaders).filter(header -> header.getName().equals("X-
Subject-Token")).collect(Collectors.toList());
            String token = collect.get(0).getValue();
            tokenCache.put(cacheKey, token);
            return token;
        }
    }

    /**
     * 推送CloudEvents事件到EG-engine
     *
     * @param cloudEvent
     * @throws Exception
     */
    public void publish(CloudEvent cloudEvent) throws Exception {
        String body = buildBody(cloudEvent);
        Map<String, String> headers = new HashMap<>();
        headers.put("X-Auth-Token", getToken(DOMAIN_NAME, NAME));

        HttpEntity resEntity = getPostResponse(URL, headers, body).getEntity();
        if (resEntity != null) {
            System.out.println(System.getProperty("line.separator") + EntityUtils.toString(resEntity, "UTF-8"));
        }
    }

    /**
     * 构建CloudEvents实体
     *
     * @param source 事件源名称
     * @return CloudEvents实体
     */

    private CloudEvent buildCloudEvent(String source) {
        io.cloudevents.core.v1.CloudEventBuilder cloudEventBuilder = CloudEventBuilder.v1()
                .withId(UUID.randomUUID().toString())
                .withSource(URI.create(source))
                .withType(JsonFormat.CONTENT_TYPE)
                .withTime(OffsetDateTime.now());
        return cloudEventBuilder.build();
    }

    private static String buildBody(CloudEvent cloudEvent) {
        JsonFormat jsonFormat = ((JsonFormat) Objects.requireNonNull(EventFormatProvider.getInstance()
                .resolveFormat("application/cloudevents+json"))).withForceNonJsonDataToString();
        EventFormatProvider.getInstance().registerFormat(jsonFormat);
        return "{\"events\": [" + new String(jsonFormat.serialize(cloudEvent)) + "]}";
    }

    /**
     * 使用指定的headers和 body进行https请求
     *
     * @param headers 请求headers
     * @param body    请求体
     * @return
     * @throws Exception
     */
    private HttpResponse getPostResponse(String url, Map<String, String> headers, String body) throws
```



```
Exception {
        CloseableHttpClient client = null;
        SSLConnectionSocketFactory scsf = new SSLConnectionSocketFactory(
                SSLContexts.custom().loadTrustMaterial(null, new TrustSelfSignedStrategy()).build(),
                NoopHostnameVerifier.INSTANCE);
        client = HttpClients.custom().setSSLSocketFactory(scsf).build();
        HttpPost httpPost = new HttpPost(url);

        headers.forEach(httpPost::addHeader);
        httpPost.addHeader("Content-Type", "application/json");
        httpPost.setEntity(new StringEntity(body, "UTF-8"));

        return client.execute(httpPost);
    }
}
```

参数说明：

- NAME：IAM用户名。
- PASSWORD：IAM 用户密码。
- DOMAIN_NAME：IAM domain用户名。
- IAM_ENDPOINT：IAM的终端节点，您可以从**地区和终端节点**中查询服务的终端节点。
- PROJECT_ID：即项目ID，参考**API凭证**，获取项目ID。
- CHANNEL_ID：参考**2 收集信息**，获取通道ID。
- ENDPOINT：事件网格**访问端点**。
- SOURCE：即事件源名称，参考**2 收集信息**，获取Filter中的values值。