EventGrid

SDK Reference

Issue 01

Date 2024-01-24

Huawei Cloud Computing Technologies Co., Ltd.

Address: Huawei Cloud Data Center Jiaoxinggong Road
             Qianzhong Avenue
             Gui'an New District
             Gui Zhou 550029
             People's Republic of China

Website: https://www.huaweicloud.com/intl/en-us/

Contents

1 Overview

This document describes the open-source CloudEvents SDK.

2 Collecting Information

Before using the open-source CloudEvents SDK to publish events, obtain the following information:

Channel ID

On the EG console, choose **Event Channels** in the navigation pane. The character string under each channel name is the channel ID, as shown in the following figure.
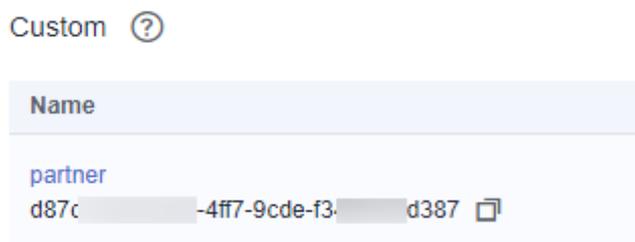
values in Filter Rule

Step 1 On the EG console, choose **Event Subscriptions** in the navigation pane, and click a subscription name to go to the details page.

Step 2 Click the event source.

Step 3 In the **Filter Rule** area, view **values**, as shown in the following figure.

* Filter Rule   ⑦   Learn how to configure a filter rule.

```
 1  {
 2    "source": [
 3      {
 4        "op": "StringIn",
 5        "values": [
 6          "test_sources_203"
 7        ]
 8      }
 9    ]
10  }
```

----End

3 CloudEvents SDK

This chapter describes how to use the open-source CloudEvents Java SDK to publish events.

Prerequisites

1. You have installed IntelliJ IDEA. If not, download it from the **IntelliJ IDEA official website** and install it.

2. Add dependencies to the **pom.xml** file. For details about how to integrate the Java SDK for API request signing, see **AK/SK Signing and Authentication Guide**.

```
<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>
```

📖 **NOTE**

Replace **cloudevents.version** with the latest version 2.2.0.

Publishing Events

The sample code for publishing events is as follows (replace the text in bold with actual values):

```
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;
```

```java
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();
        }
    }

    /**
     * Obtain a token.
     *
     * @param userName (IAM username)
     * @param domainName (Account name)
     * @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;
    }
}

/**
 * Push CloudEvents events to 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"));
    }
}

/**
 * Build a CloudEvents entity.
 *
 * @param source (Event source name)
 * @return CloudEvents entity
 */

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)) + "]}";
}

/**
 * Use the specified headers and body to send HTTPS requests.
 *
 * @param headers (request headers)
```

```
    * @param body (request body)
    * @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);
   }
}
```

Parameter description:

- **NAME**: IAM username.

- **PASSWORD**: IAM user password.

- **DOMAIN_NAME**: Account name.

- **IAM_ENDPOINT**: IAM endpoint. For details, see **Regions and Endpoints**.

- **PROJECT_ID**: The project ID. Obtain it by referring to **API Credentials**.

- **CHANNEL_ID**: Obtain a channel ID by referring to **2 Collecting Information**.

- **ENDPOINT**: EG **endpoint**.

- **SOURCE**: Event source name, which can be obtained from **values** in **Filter Rule** by referring to **2 Collecting Information**.