Evergen OEM-API Connect: User Guide for Connecting

This document provides instructions for an Evergen OEM-API Connect user to connect to the API, and send a sample telemetry package to confirm connectivity. This document provides sample Java code for implementing an SQS client application and running this within an AWS EC2 instance, providing the necessary permissions in the user's AWS account.

Note: This documentation summarises key parts of existing AWS documentation. Links to the original documentation will be provided in each step as a more comprehensive reference.

Steps:

1. Create IAM Policy
2. Create IAM Role and share IAM role ARN with Evergen
3. Create EC2 Instance
4. Attribute IAM Role to EC2 instance
5. Run Sample Java application in EC2 instance to test connectivi

1. Create IAM Policy

- Inside the AWS console, navigate to: *IAM→Policies→Create policy*
- Create a Policy with the *JSON* config, copying the below JSON into the *Policy editor*,

- Add a name and description, and confirm the creation of the policy with all other default settings
- AWS documentation

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:SendMessage",
                "sqs:GetQueueAttributes",
                "sqs:ChangeMessageVisibility"
            ],
            "Resource": "arn:aws:sqs:ap-southeast-2:406871981
087:telemetry-ADD-YOUR-VENDOR-dev"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sqs:GetQueueUrl",
                "sqs:ChangeMessageVisibility",
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage",
                "sqs:GetQueueAttributes"
            ],
            "Resource": "arn:aws:sqs:ap-southeast-2:406871981
087:commands-ADD-YOUR-VENDOR-dev"
        }
    ]
}
```

2. Create IAM Role or IAM User and share IAM role/user ARN with Evergen

- Create IAM Role → For running your application in AWS
  - Inside the AWS console, navigate to: *IAM→Roles→Create role*
  - For the ***Trusted entity type*** select ***AWS service***
  - For the ***Use case,*** select ***EC2/EC2 (Allows EC2 instances to call AWS services on your behalf.)***
  - For **Permissions policies** select the policy created in Step 1.
  - Add a name and description, and confirm the creation of the role with all other default settings
  - Select this role from the list to view its Role summary, and share the listed ARN value with Evergen. This should be of the format: *arn:aws:iam::123456789012:role/iam-role-name.*
  - Await Evergen's response with the name of the queue created for you.
  - AWS documentation
- Create IAM User → For running your application outside of AWS
  - Inside the AWS console, navigate to: *IAM→Users→Create user*
  - Provide a ***User name***
  - When you ***Set Permissions***, select ***Attach policies directly***, then search and select the IAM Policy created in Step 1.
  - Confirm creation of the user
  - Select this user from the list to view its User details, and share the listed ARN value with Evergen. This should be of the format: *arn:aws:iam::123456789012:user/iam-user-name.*
  - Create an access key for your IAM user by selecting: ***Create access key*** inside the ***User details***
  - Select the desired use case of the access key, E.g: ***Local code***, ***Application running outside AWS*** etc.
  - Set a ***Description tag value***, confirm ***Create access key*** and save the **Access key** and **Secret access key**

- Await Evergen's response with the name of the queue created for you.
- Retrieve your Inside the code example, make the following change:

```
// Instead of using the default credentials, E.g:

 AmazonSQS sqsTelemetryClient = AmazonSQSClientBuilder.
standard()
 .withRegion("ap-southeast-2") // Replace with your des
ired AWS region
 .build();

// Change this to use specific credentials, E.g:
String accessKey = "YOU_ACCESS_KEY";
String secretKey = "your-secret-key";

BasicAWSCredentials awsCredentials = new BasicAWSCreden
tials(accessKey, secretKey);

AmazonSQS sqsTelemetryClient = AmazonSQSClient.builder
()
    .withCredentials(new AWSStaticCredentialsProvider(a
wsCredentials))
    .withRegion("ap-southeast-2")
    .build();
```

3. Create EC2 Instance

- Inside the AWS console, navigate to: EC2→*Instances*→*Launch an instance*
- Select ***Amazon Linux*** as the **Amazon Machine Image**
- Select existing or ***Create new key pair***
- Use all other default settings and confirm ***Launch Instance***
- AWS documentation

4. Attribute IAM Role to EC2 instance

- Select your new instance from the ***Instances*** list, and inside the **Instance summary** select: ***Actions→Security→Modify IAM role***
- Select the IAM role from Step 2., and proceed with ***Update IAM role***
- AWS documentation

5. Run Sample Java application in EC2 instance to test connectivity

- In the sample application App.java file, line 79, change the value of the `queueUrl` variable to be the URL shared with you by Evergen in Step 2.
- In the sample application App.java file, line 119, update the source from http://example.com/source to your company
- Inside instance summary page select ***Connect→SSH client***, retrieve the instance address of the format: `your-instance` `@ec1-23-45-67-890.compute-1.amazonaws.com` `:/home/ec2-user`
- (Inside your local terminal) Copy the sample java application to the EC2 instance using SCP (Secure Copy Protocol), run: (updating the instance address and file path to .pem key files and sample java app)

```
scp -i "/path-to-you-key-pair-pem-file/key.pem" \
  -r /path-to-sample-folder/sample-java-app-folder \
  your-instance@ec1-23-45-67-890.compute-1.amazonaws.com:/
home/ec2-user
```

- (Inside your local terminal) Connect to the EC2 instance using SSH, run : (updating the instance address and file path to .pem key files)

```
ssh -i "/path-to-you-key-pair-pem-file/key.pem"` \
  your-instance@ec1-23-45-67-890.compute-1.amazonaws.com
```

- (Inside your SSH session connected to the EC2 instance) Install Java on your instance, run:

```
sudo yum install java-21
```

- (Inside your SSH session connected to the EC2 instance) Install Maven, run:

```
sudo wget https://repos.fedorapeople.org/repos/dchen/apach
e-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-ap
ache-maven.repo
sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apach
e-maven.repo
sudo yum install -y apache-maven
```

- (Inside your SSH session connected to the EC2 instance) Package the executable and run the sample application:

```
mvn clean package
mvn exec:java -Dexec.mainClass=org.example.sqs.App
```

- Expected output of the demo should resemble that below, and check with the Evergen team on Slack or email that your sample telemetry was received.

```
# Expected output:

CloudEvent: {"specversion":"1.0","id":"12345","source":"htt
p://example.com/source","type":"example.event.type","datacont
enttype":"application/json","data":{"siteID":"Site125","batte
ryInverters":null,"hybridInverters":[{"deviceID":"Device12
3","deviceTime":null,"batteryPowerW":0,"meterPowerW":0,"solar
PowerW":0,"batteryReactivePowerVar":0,"meterReactivePowerVa
r":0,"gridVoltage1V":0.0,"gridVoltage2V":0.0,"gridVoltage3V":
0.0,"gridFrequencyHz":0.0,"cumulativeBatteryChargeEnergyWh":
0.0,"cumulativeBatteryDischargeEnergyWh":0.0,"stateOfCharge":
0.0,"stateOfHealth":0.0,"maxChargePowerW":0,"maxDischargePowe
rW":0}],"solarInverters":null,"meters":null}}
Message sent with ID: your-message-id
```