FAQ COMBIVIS Connect

How To Connect to AWS IoT

Part	Version	Revision	Date	Status
en	5.1.1183.81	001	2021-05-04	Released

Content

Introduction **2**

Configure AWS **2**
1. Set Policies and Roles . . . 2
2. Register device to AWS IoT . . . 4
3. Configure Thing Policy . . . 8
4. Create AWS IoT Rule . . . 10
5. Configure Lambda Function . . . 12
6. Create DynamoDB Table . . . 14

Configure HMI Project **15**
1. Data Logger Configuration . . . 15
2. Cloud Push Agent Configuration . . . 17
3. Cloud Runtime Configuration . . . 17

Transfer Security Certificates and Upload HMI Project **19**
1. Transfer Security Certificates . . . 19
2. Upload HMI Project . . . 20

Test Connection **22**
1. Log Data . . . 22
2. Test Debug Screen . . . 23
3. Test Debug Window . . . 24
4. Cloud Watch Log . . . 25
5. DynamoDB Table . . . 25

Disclaimer **27**

Introduction

This document explains how to push data to Amazon Web Services (AWS). The following are required for this FAQ:

- C6 Router with Cloud Runtime
- Combivis Connect Domain
- Combivis Studio HMI
- AWS Account

Configure AWS

This section will review how to configure the required AWS services to push data from the C6 Router to AWS.

1. Set Policies and Roles

AWS requires policies and roles to control what a user or program can do. When deploying to production these policies and roles should be reviewed and modified for increased security and protection if necessary.

The following user policies are required to access the AWS services required in this FAQ. Policies and roles are configured in the Identity and Access Management (IAM) in AWS.

- **User Policies**
  - AWSLambdaFullAccess
  - AWSIoTFullAccess
  - AmazonDynamoDBFullAccess
  - CloudWatchLogsFullAccess (to enable debug only)

This FAQ will require a role for the execution of a lambda function. To create a new role select Roles under Access Management in the IAM console and Create role.

Next, choose AWS service and Lambda for the use case and select Next.

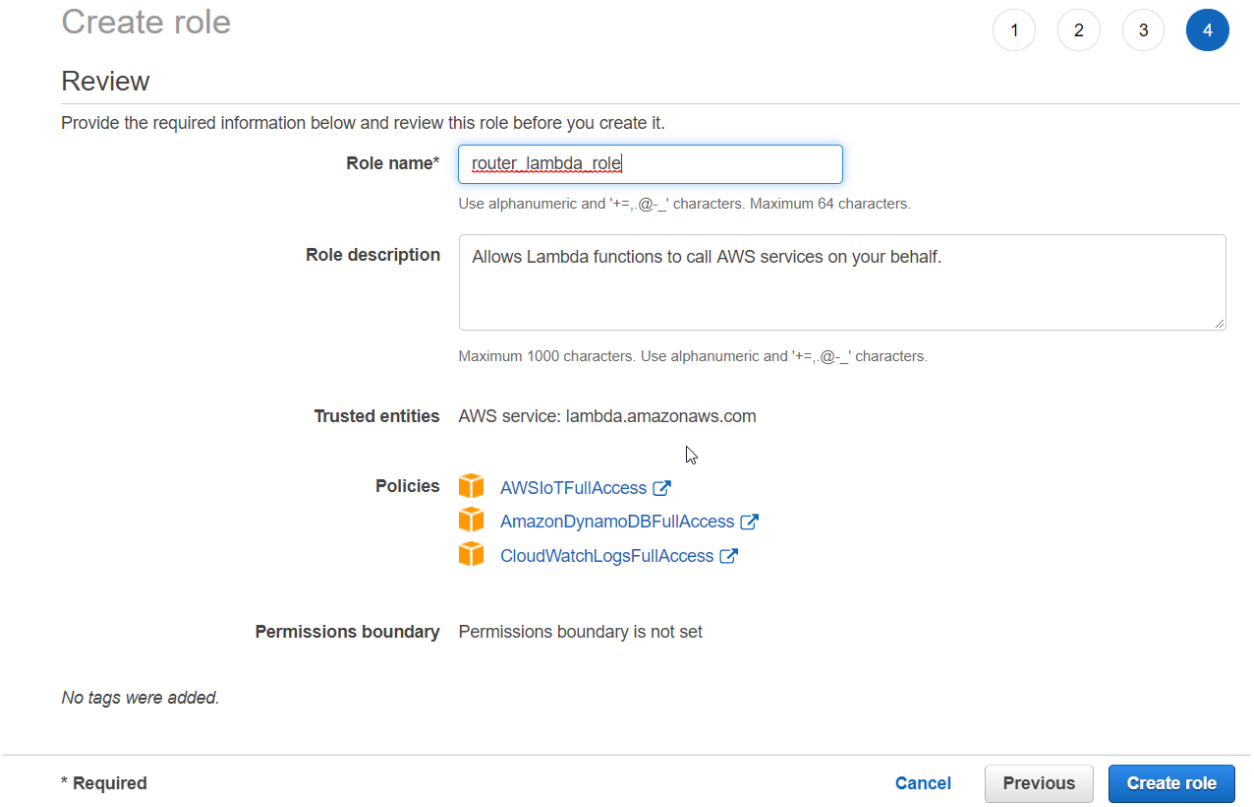

Fig. 1: Create IAM Role

The following policies are required for this project.

- AWSIoTFullAccess
- AmazonDynamoDBFullAccess
- CloudWatchLogsFullAccess (to enable debug only)

Search for each policy and check the box to the left of the policy name to add the policy to the role. Select Next:tags after the required policies have been selected. Setting a tag for this role is optional, for this FAQ a tag will not be applied to this role. Finally, select next and create.

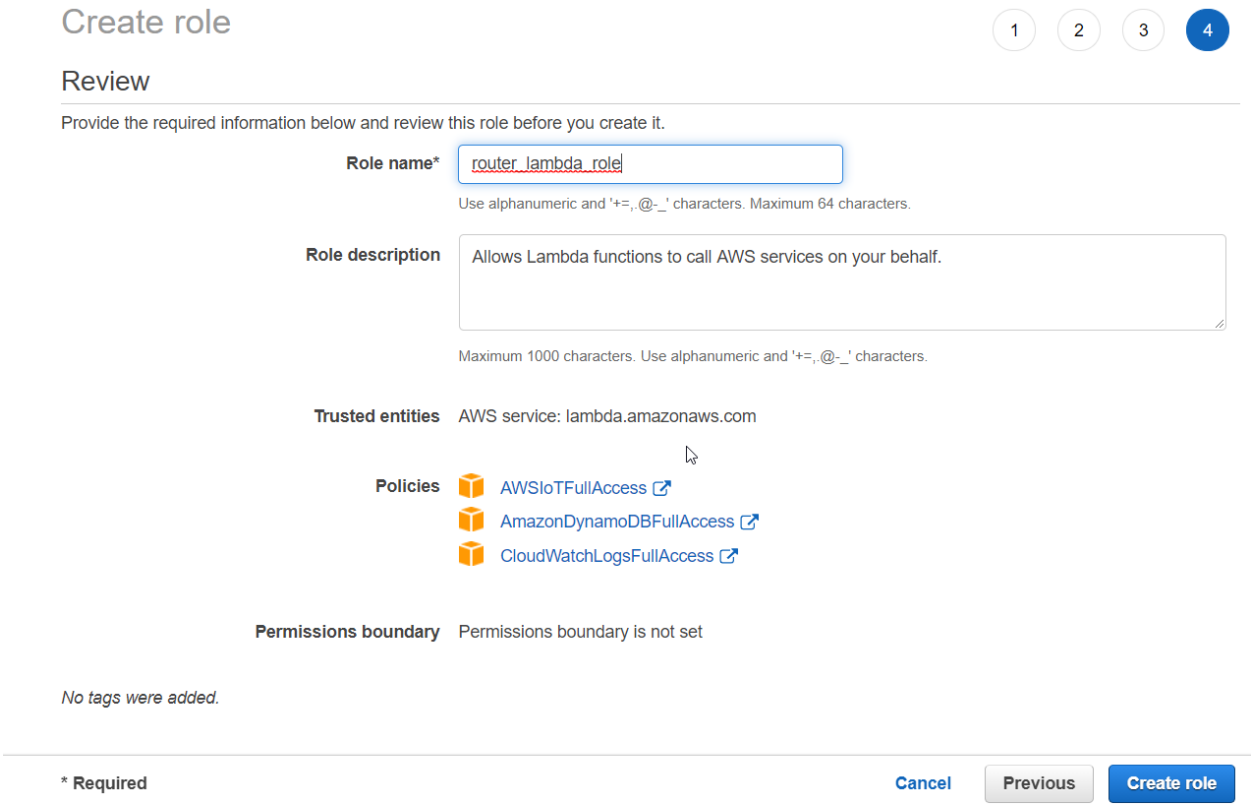

Fig. 2: Configure IAM Role

2. Register device to AWS IoT

Navigate to the AWS IoT service and select Onboard -> Get started.

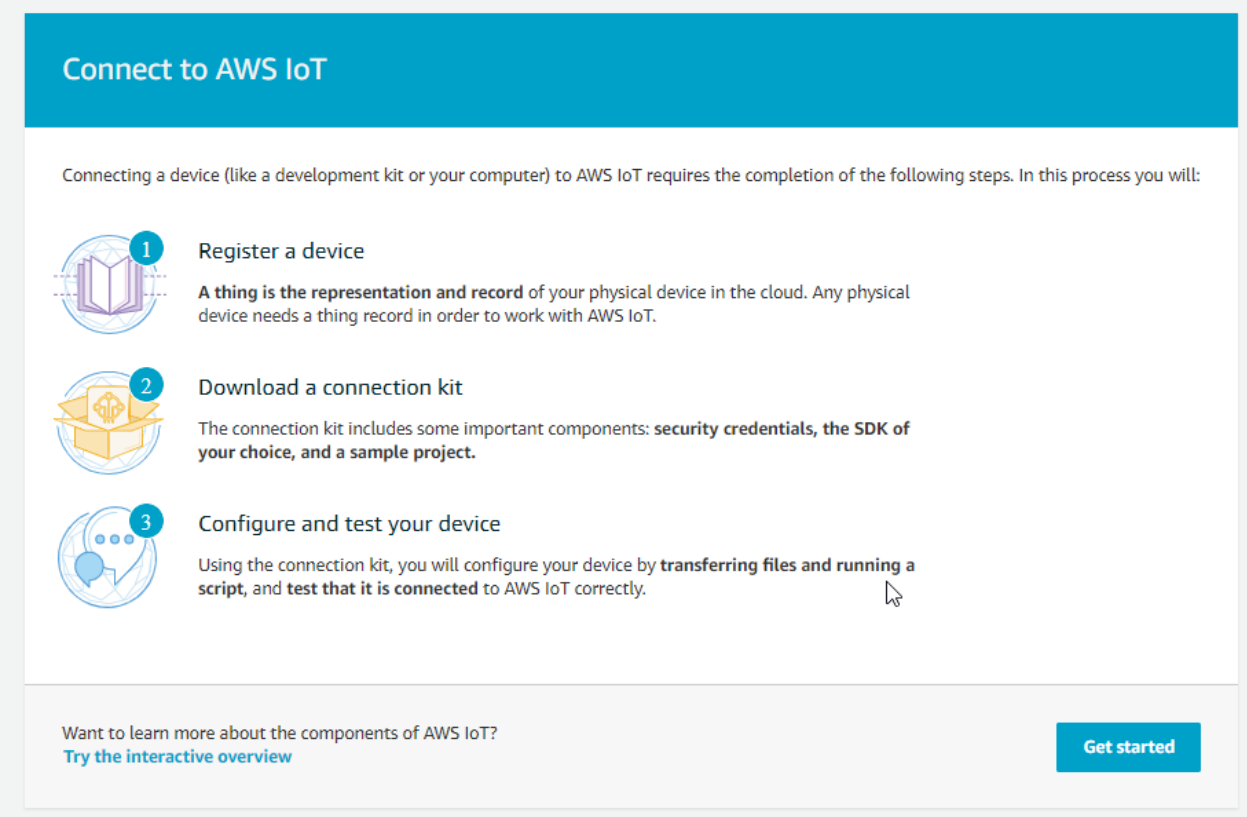

Fig. 3: Connect Device

Next, select the platform and SDK for connecting to AWS IoT. Select Windows -> Node.js -> Next.

Devices in AWS are represented as Things. Give the C6 Router a Thing name.

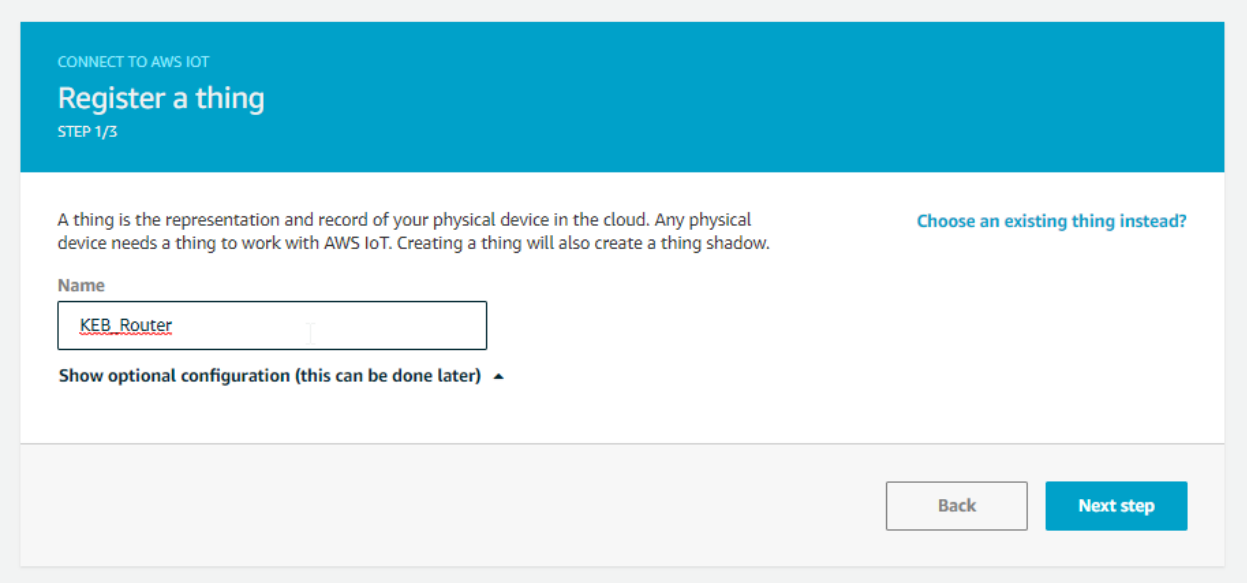

Fig. 4: Register Thing

After defining the thing name, download the connection kit. The connection kit will contain security certificates that will need to be transferred to the C6 Router.

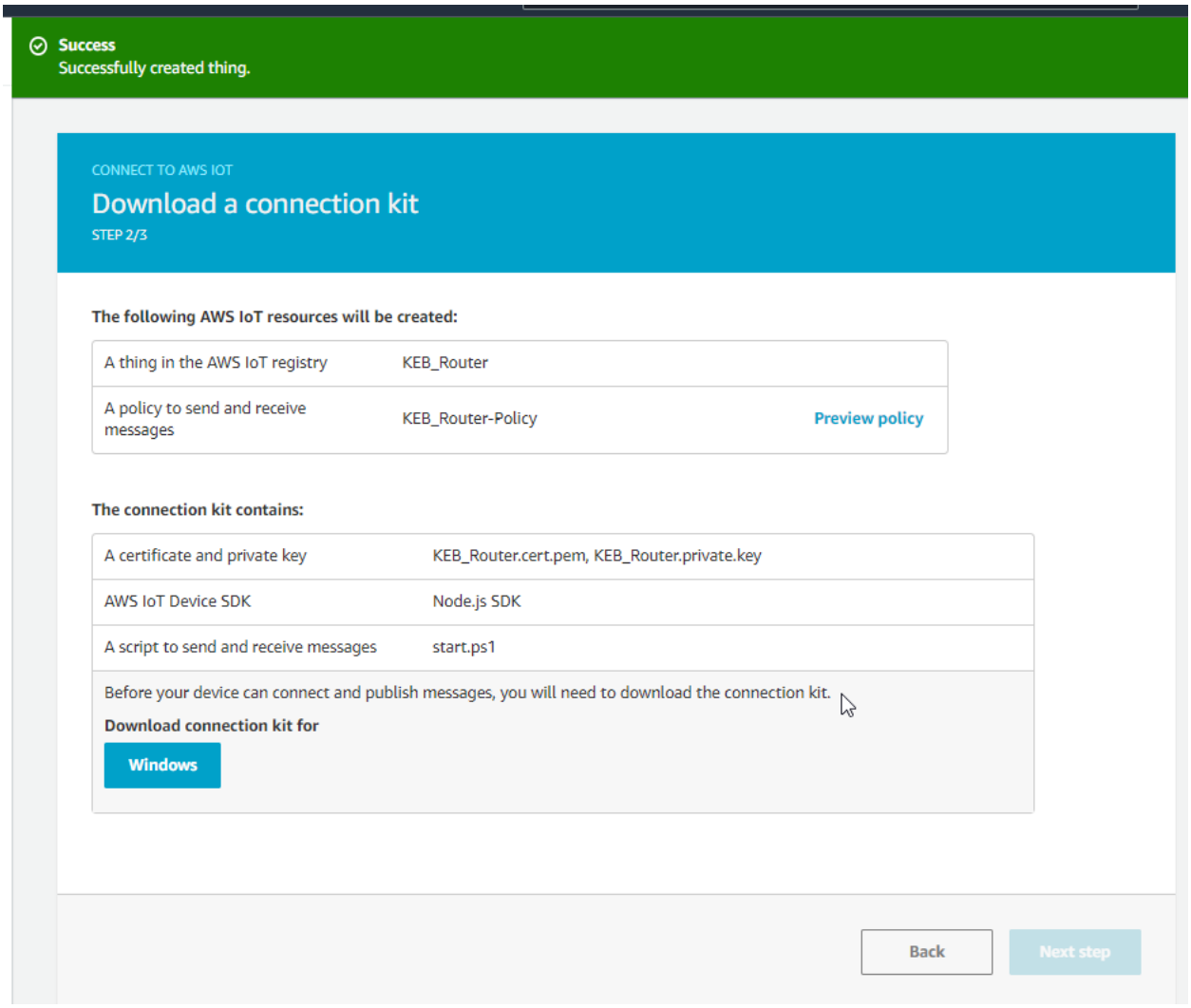

Fig. 5: Download Connection Kit

Finally, select Done. At this point the C6 Router has been configured to AWS IoT and the security certificates required to push data have been downloaded.

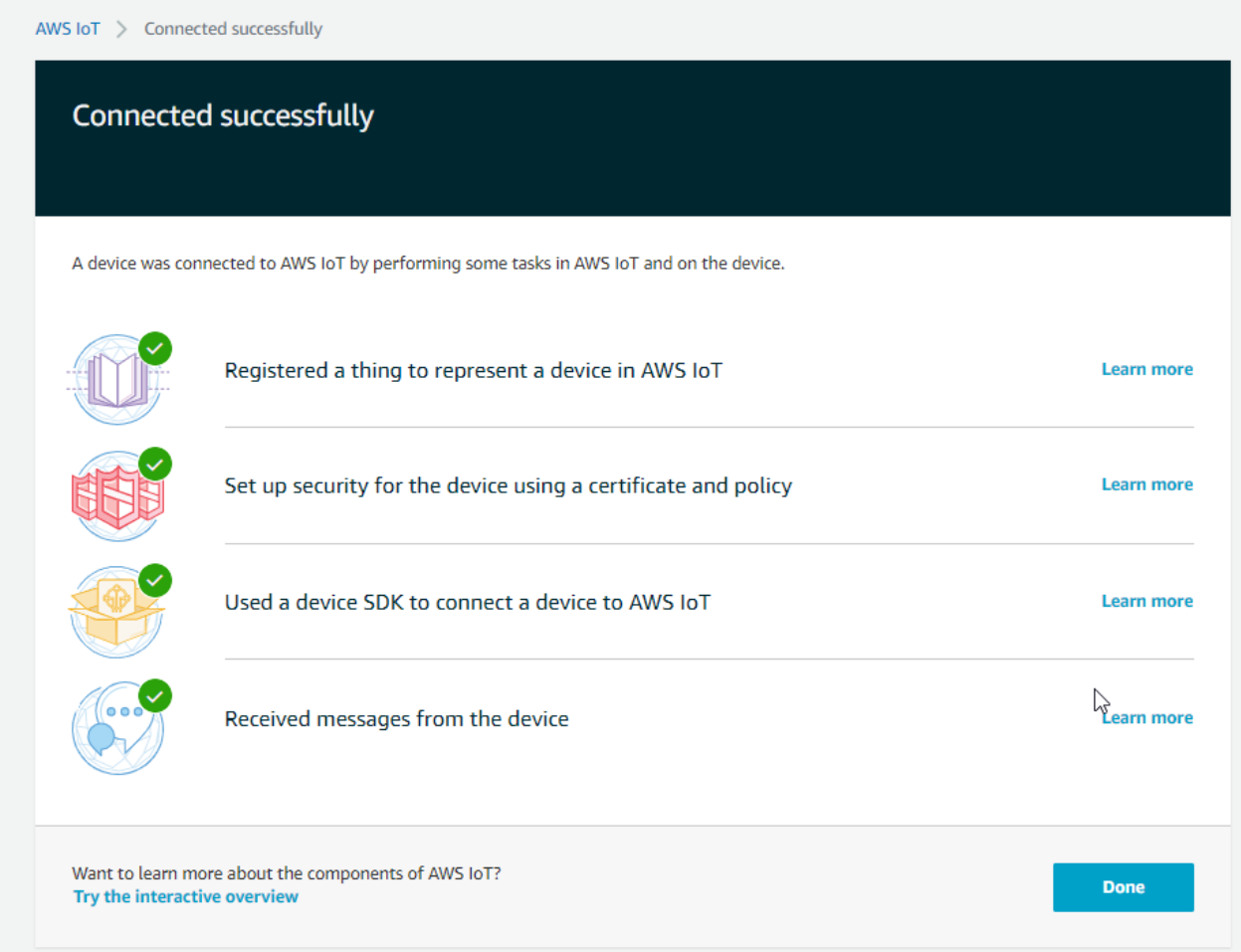

Fig. 6: Device Succesfully registered to AWS IoT

3. Configure Thing Policy

The security policy of the thing created will need to be edited to allow for access to resources within the AWS IoT. To edit the security policy, select Secure -> Policies and thing name policy.

The C6 Router will push to the topic raw, and the router will need to subscribe and connect to AWS IoT. Add the topic raw under the iot:Publish and iot:Recieve action. Next, add the think name under the iot:Subscribe and iot:Connect actions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:us-east-2:952356642709:topic/sdk/test/java",
        "arn:aws:iot:us-east-2:952356642709:topic/sdk/test/Python",
        "arn:aws:iot:us-east-2:952356642709:topic/topic_1",
        "arn:aws:iot:us-east-2:952356642709:topic/topic_2",
        "arn:aws:iot:us-east-2:952356642709:topic/raw"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:us-east-2:952356642709:topicfilter/sdk/test/java",
        "arn:aws:iot:us-east-2:952356642709:topicfilter/sdk/test/Python",
        "arn:aws:iot:us-east-2:952356642709:topicfilter/topic_1",
        "arn:aws:iot:us-east-2:952356642709:topicfilter/topic_2",
        "arn:aws:iot:us-east-2:952356642709:topicfilter/E4_Router"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:us-east-2:952356642709:client/sdk-java",
        "arn:aws:iot:us-east-2:952356642709:client/basicPubSub",
        "arn:aws:iot:us-east-2:952356642709:client/sdk-nodejs-*",
        "arn:aws:iot:us-east-2:952356642709:client/E4_Router"
      ]
    }
  ]
}
```

Fig. 7: Edit security policy

4. Create AWS IoT Rule

The C6 Router pushes messages to AWS IoT as binary packets. A rule must be created to decode the binary packets into JSON using AWS Lambda. The rule will trigger AWS Lambda to process each packet recieved from AWS IoT.

From the AWS IoT console select Act -> Create Rule. Next, choose a name for the rule and using SQL version 2016-03-23 write a query that will trigger the rule. The following query can be used: SELECT encode(*,'base64') as payload from 'raw'

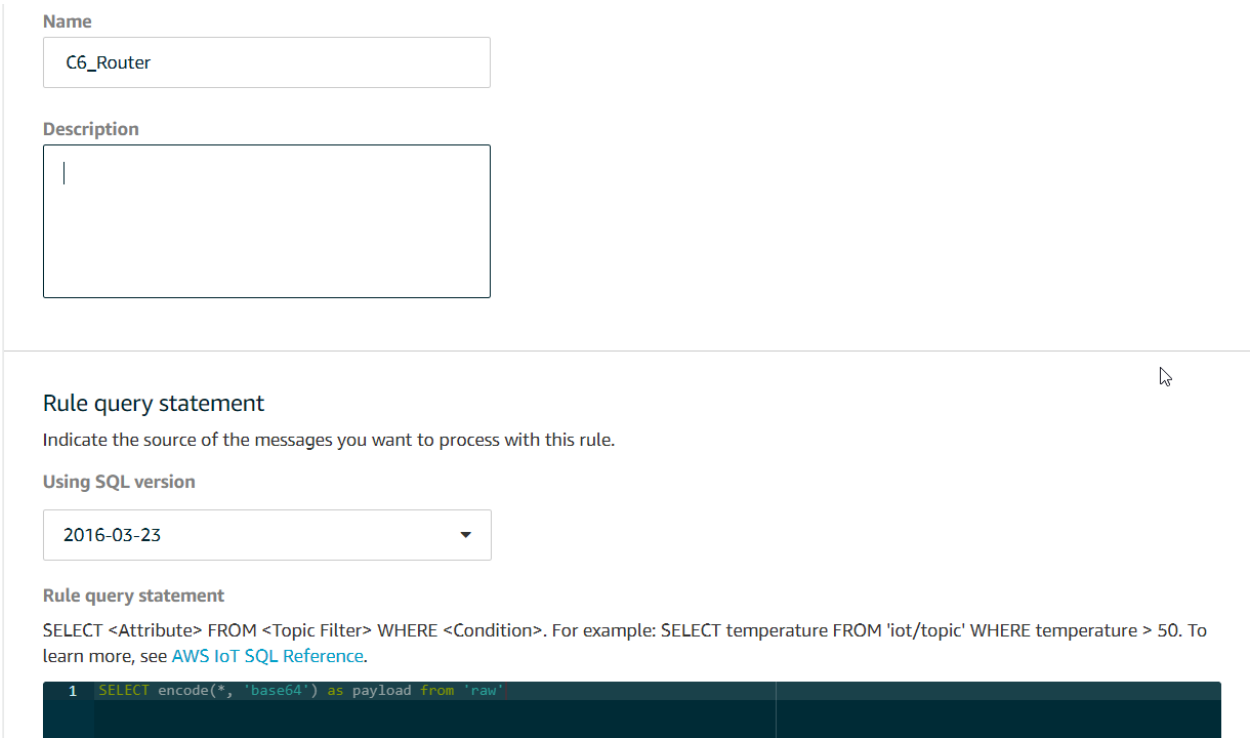

Fig. 8: Create AWS IoT Rule

Next, configure the action for the rule. Select Add Action -> Send a message to a Lambda function -> Configure action.

Select create a new Lambda function and give the function a name and select the Node.js 10.x runtime. Finally, select create function.

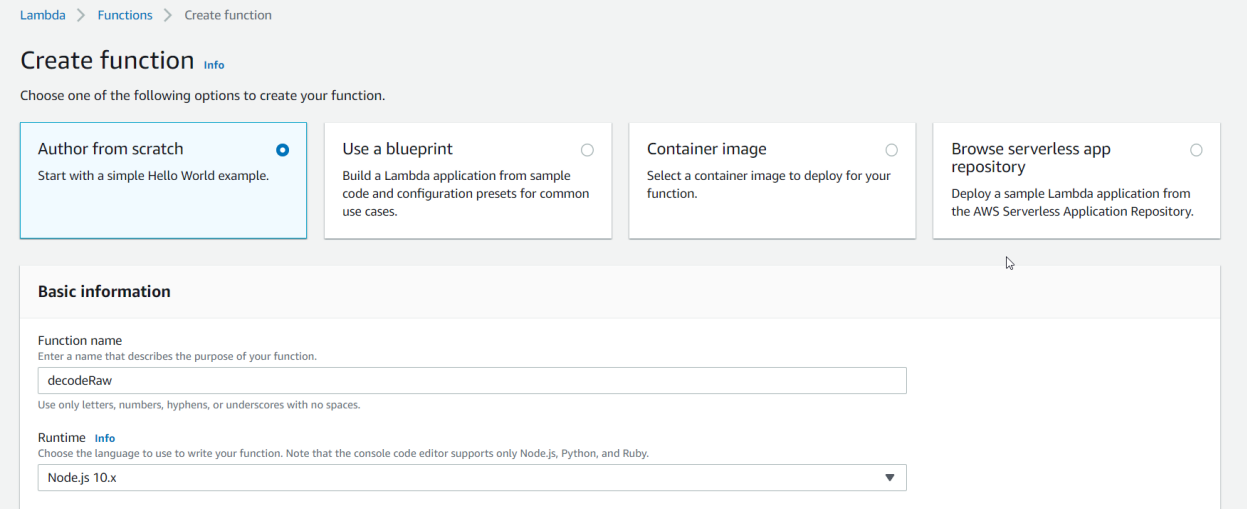

Fig. 9: Create Lambda Function

After creating the lambda function navigate back to the AWS IoT rule and select the name of the lambda function previously created. Next, select create rule.

The AWS IoT rule is now setup and ready to trigger a lambda function.

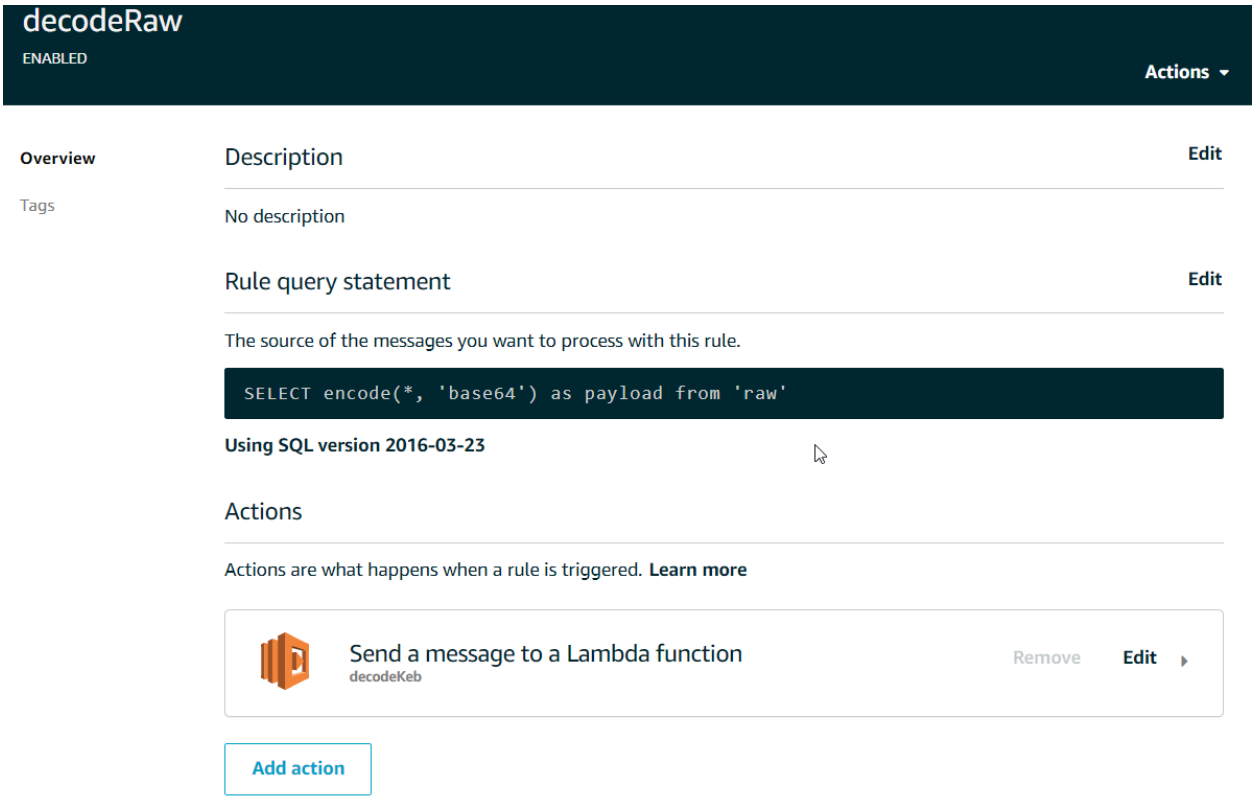

Fig. 10: AWS IoT rule created

5. Configure Lambda Function

The AWS Lambda service will decode binary packets into JSON and can be saved into a database like DynamoDB or MySQL. This FAQ will save the decoded packets into DynamoDB.

Sample code for decoding packets has been supplied with this FAQ. Open AWS Lambda service and select the lambda function created in the previous step.

Next, paste the sample code into the lambda code editor.

Select Manage -> Things -> Select Thing name. Select Interact and copy the HTTPS endpoint. The variable ioTEndpoint can be set as the HTTPS endpoint.

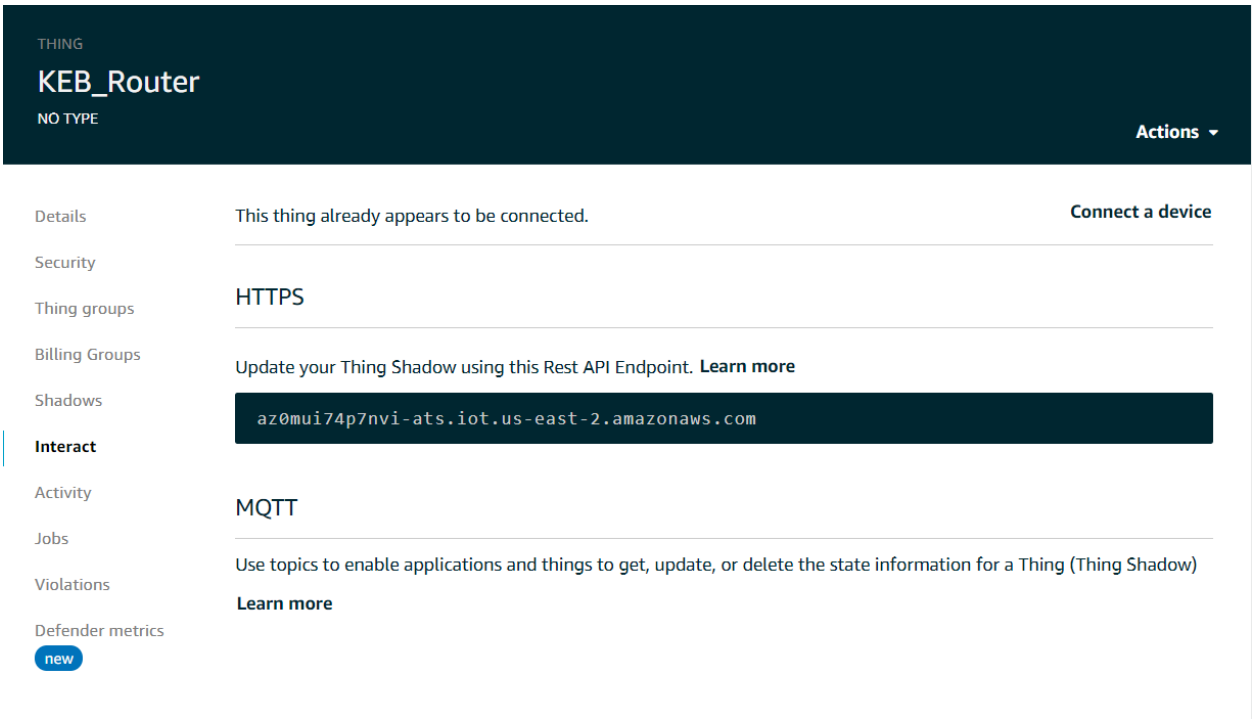

Fig. 11: Thing endpoint

Lastly, set the dynamodbTable variable equal to a name and select Deploy. The DynamoDB table will be created in the next step.

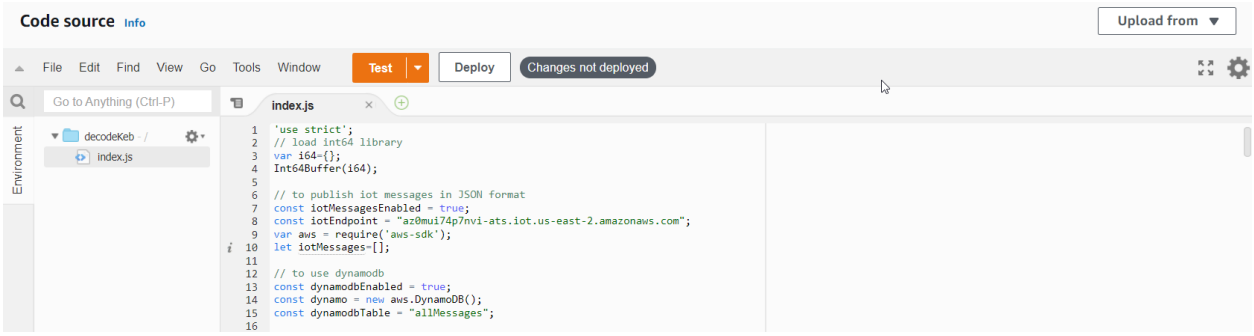

Fig. 12: Lambda function configuration

6. Create DynamoDB Table

The decoded binary packets from the Lambda function configured in the previous step will be saved into DynamoDB.

To create a DynamoDB table open the DynamoDB service and select Create Table. Set the table name used in configuring the lambda function. Next, set the Primary key as timestamp of type Number. Select Add sort key and set it as tag of type String. Finally, select Create and the table will be created.

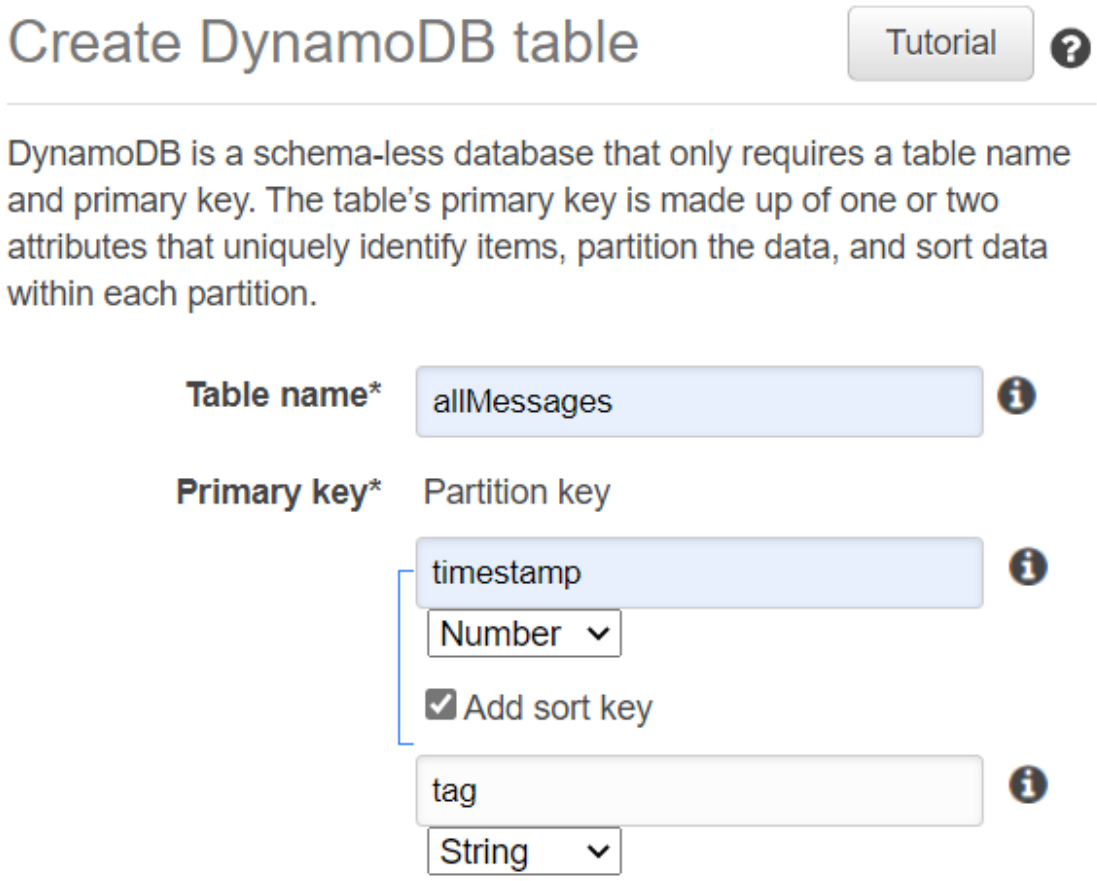

Fig. 13: DynamoDB table creation

Configure HMI Project

A Combivis Studio HMI project must be created to push data to AWS. A sample project has been created for this FAQ and can be used for initial testing. The critical HMI resources used will be reviewed in this section.

1. Data Logger Configuration

The Data Logger resource is used to push data to the cloud. Tags are configured to the data logger and recorded into the database.

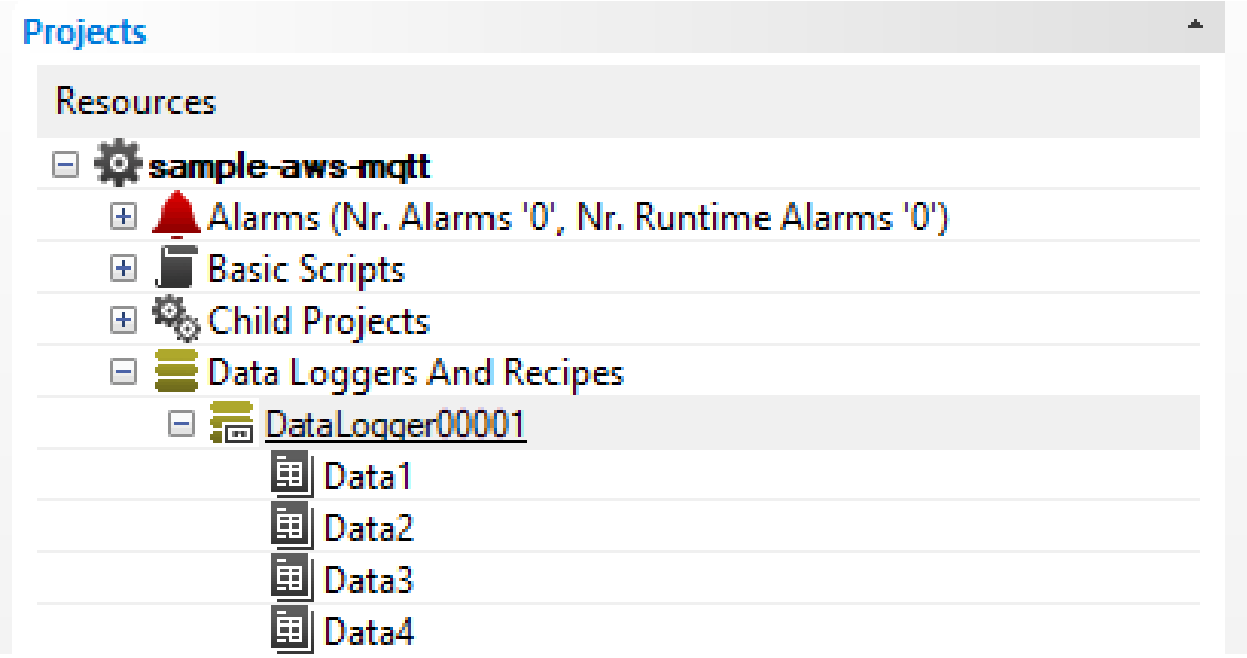

Fig. 14: Data Logger

The style property defines how the data logger records entries into the database. If the setting Records on Change is selected the datalogger will add entries into the database anytime a tag within the datalogger changes its value. The setting Record on Command adds entries into the database anytime the Recording Variable is strobed up in the Execution properties.

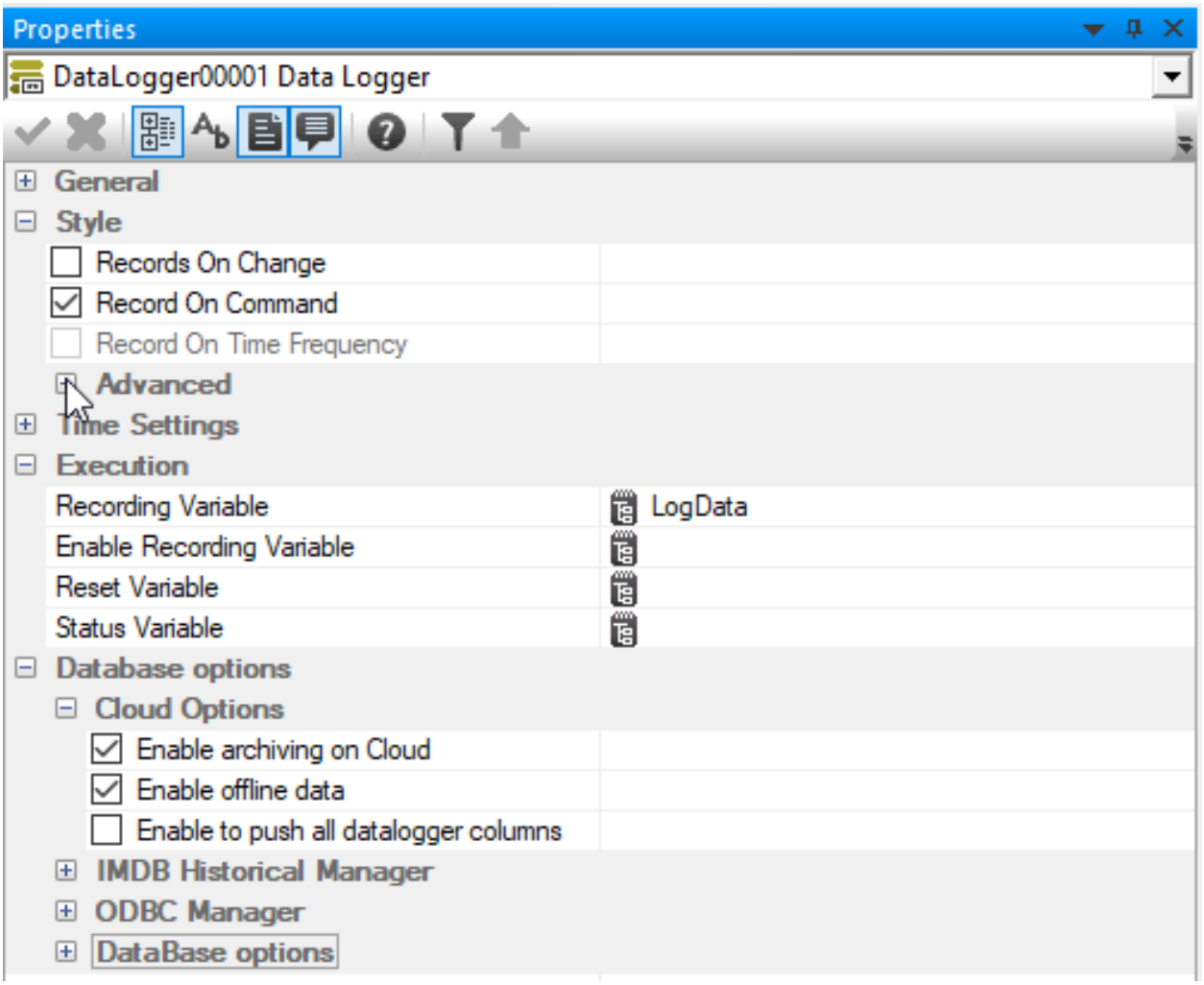

Fig. 15: Data Logger Configuration

Lastly, the Cloud Options need to be configured in the Database options properties. The Enable archiving on Cloud property is required to push data to the cloud. The Enable offline data propery will save data locally to the router in the event of internet loss.

2. Cloud Push Agent Configuration

The frequency in which the router will push data to the cloud is configured using the Publish Period setting in the Cloud Push Agent Configuration properties. The Cloud Push Agent Configuration properties can be found in the project properties.

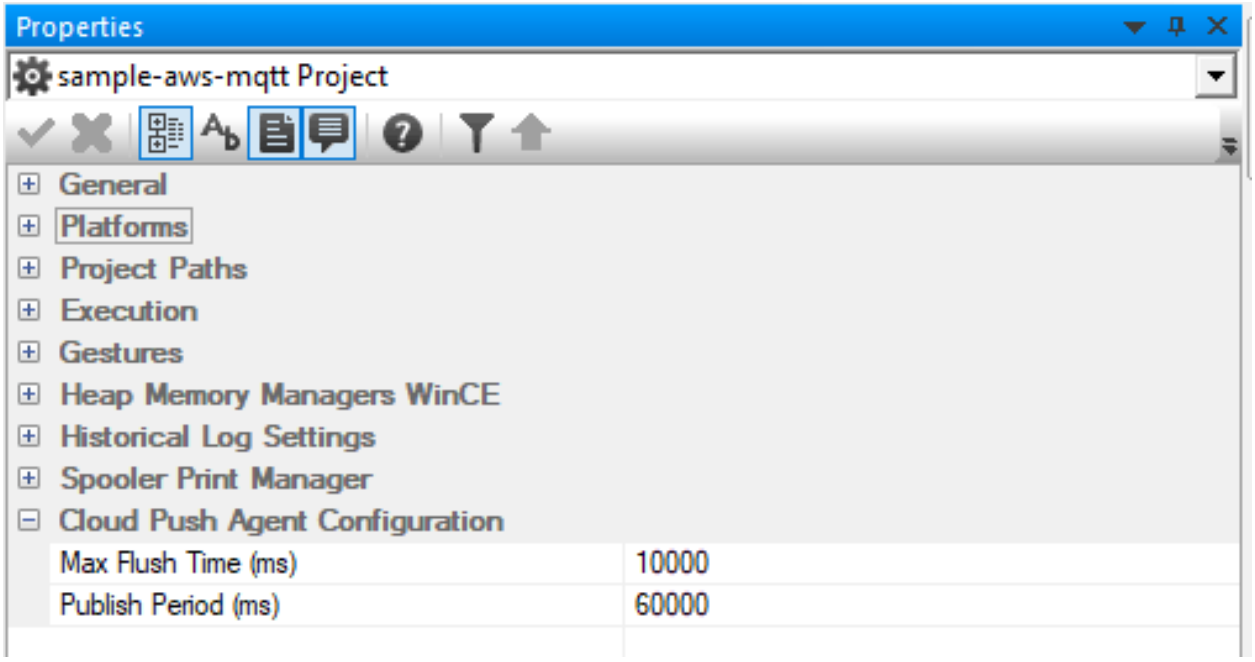

Fig. 16: Cloud push agent configuration

3. Cloud Runtime Configuration

A configuration file is used to specify parameters for connecting to AWS IoT. The configuration file is created automatically and is located in the RESOURCES folder of the HMI project.

The following parameters are required to connect to AWS IoT, to set a parameter value uncomment the desired tag and add a value between the two tags.

- UseManualAuthenticationMode: This must forces the cloud runtime to push data to a 3rd party cloud service. This must be set to TRUE.
- DeviceID: This represents the device which is sending the data. You can configure multiple devices to send data to the same thing.
- ClientID: This identifies the client in the MQTT protocol. It can be the same as the DeviceID.

- Endpoint: This is the url where the data will be pushed.
- Protocol: Only MQTT is supported.
- QoS: The MQTT QoS levels 0 and 1 are implemented. Level 0 means that there are no guarantees of delivery of the messages. Level 1 means that every message will be delivered at least one time, with possible duplication
- Target: This is the name of the topic where the binary messages will be published. The AWS Iot Rule will be bound to this topic. A good name could be raw, to indicate that the messages received here are not processed yet.
- UseSSL: This must be set to true for AWS connection.
- ClientCertificateFilePath: This is the absolute path of the client certificate received from AWS, usually with a name like ThingName.cert.pem.
- ClientKeyFilePath: This is the absolute path of the private key received from AWS, usually with a name like ThingName.private.key.
- TrustStoreFilePath: This is the absolute path of the public server identity certificate, downloadable from Verisign.
- TlsVersion: The version of transport encryption to use. Should be set to 12.

During inital connection and for troubleshooting it is often helpful to enable a debug window for additional error information. To enable the debug window uncomment the tags inside of <LoggerConfigOptions></LoggerConfigOptions>.

Transfer Security Certificates and Upload HMI Project

During this section the HMI project and required security certificates will be transferred to the router.

1. Transfer Security Certificates

The security certificates downloaded after the router was registered to AWS IoT (see *2. Register device to AWS IoT*) can now be transferred to router. The path must match the configuration file.

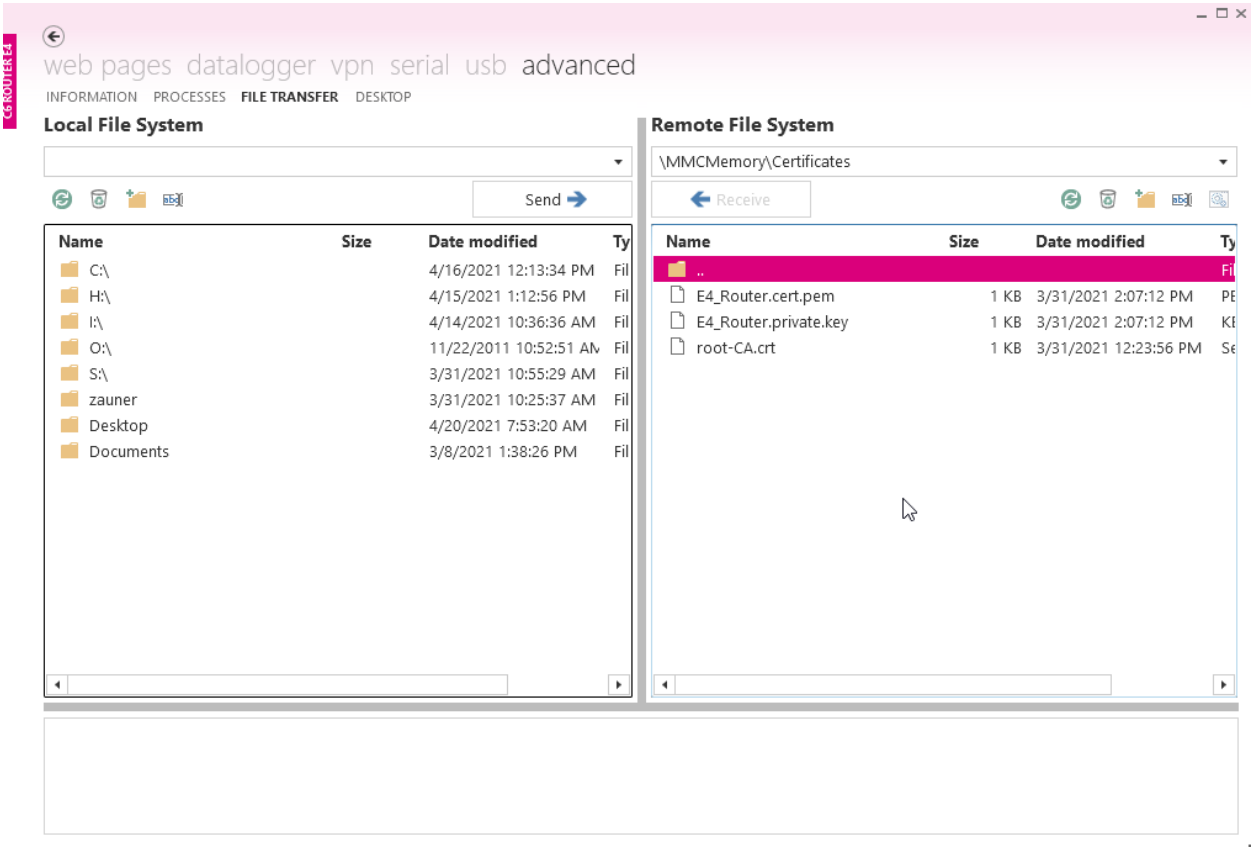

Fig. 17: Transfer Security Certificates

2. Upload HMI Project

To upload the HMI project, launch the upload window by right-clicking on the project name and select Upload Project to Device/FTP.

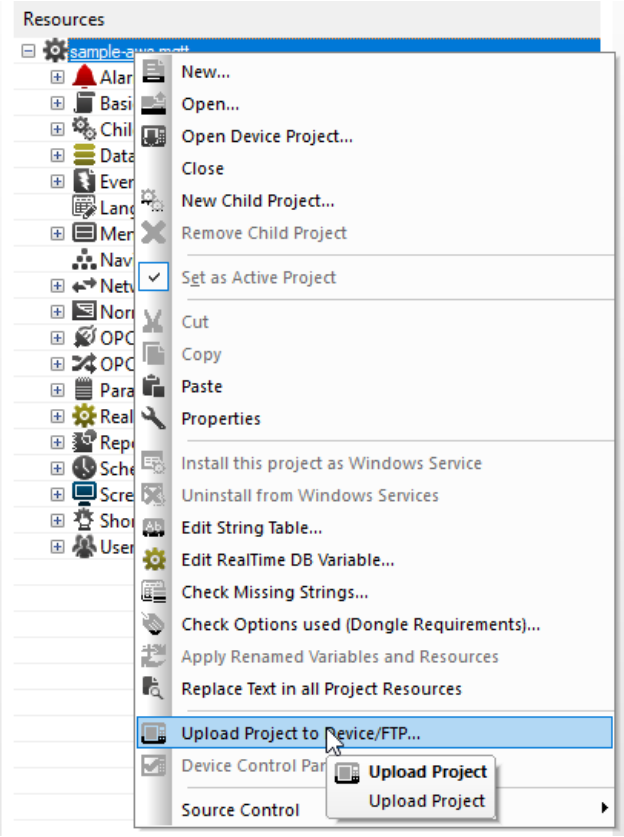

Fig. 18: Open Upload Window

After the upload window is open, enter the IP address of the router in the server field. Next, enter a directory to upload the HMI project. The project should be uploaded to the MMCMemory. To create a new folder in the MMCMemory, enter name of the new folder to the MMCMemory directory (ie. /MMCMemory/NewFolder/).

Finally, select upload project. Once the project has completed uploading select Start device project and the project will open on the router.

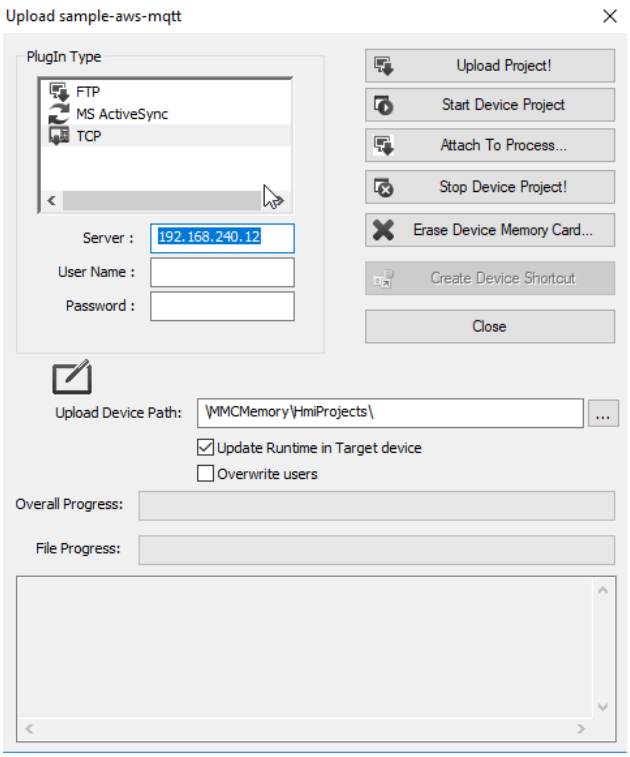

Fig. 19: Upload Project to Router

Test Connection

This section will review how to navigate the sample project provided with this FAQ as well as trouble shooting tips.

1. Log Data

After starting the HMI project, data can be logged using the LOG DATA button. Changing values of each tag can be done using the edit boxes for each respective tag.

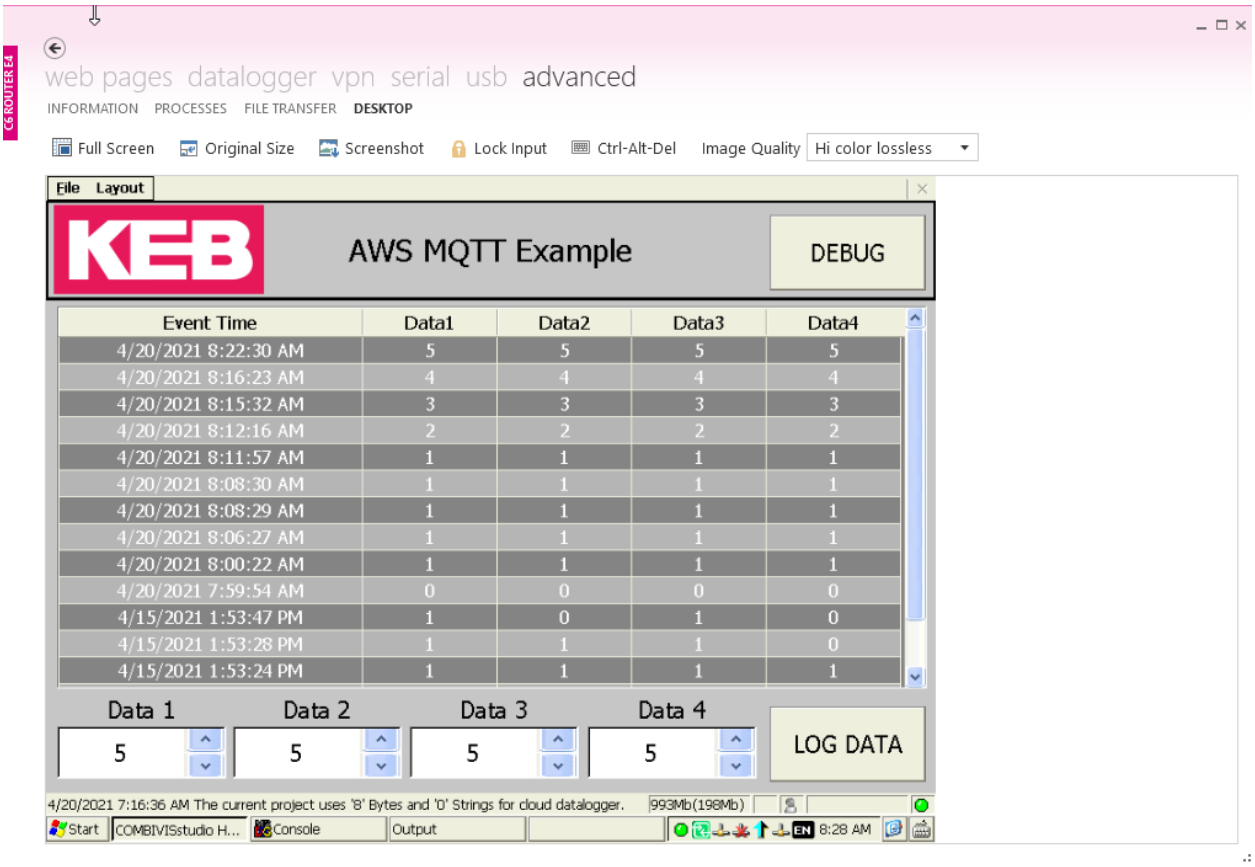

Fig. 20: Sample project data screen

2. Test Debug Screen

For trouble shooting the debug screen has been created to display the status of the connection and how many packets have been pushed since the project started.

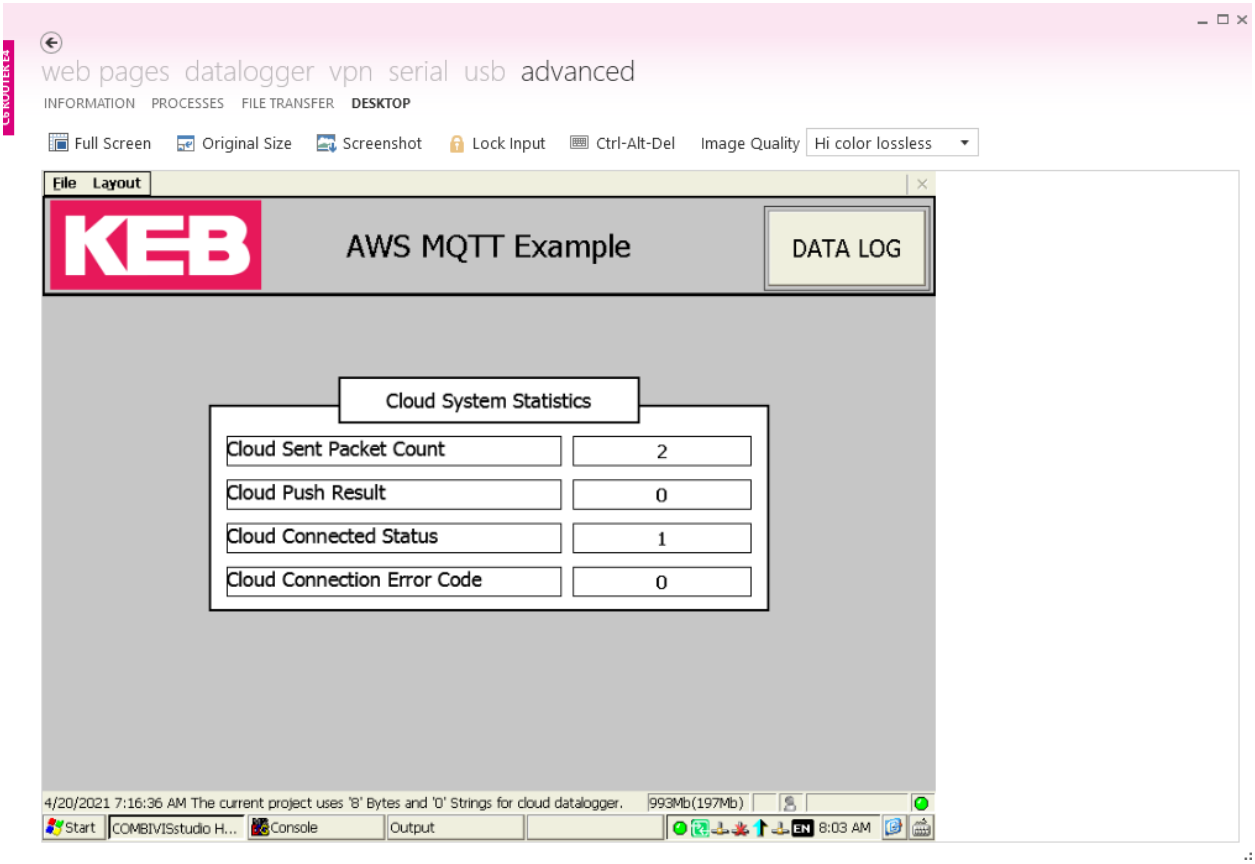

Fig. 21: Sample project debug screen

3. Test Debug Window

If the debugger window has been enabled in the CloudRuntimeConfig.xml, it can be viewed by selected the console tab on the task manager.

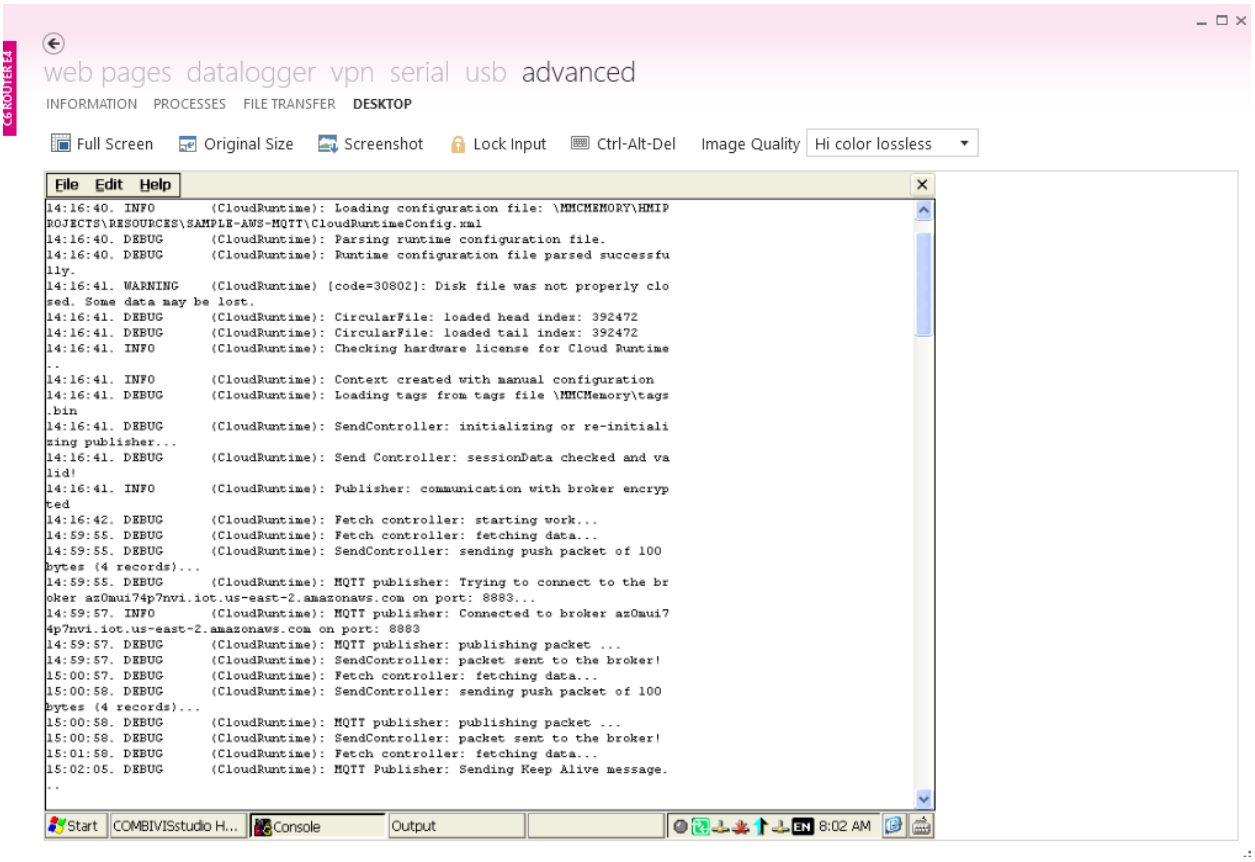

Fig. 22: Cloud debug window

4. Cloud Watch Log

The Cloud Watch Logs available in AWS can be used to debug issues within the AWS console. Common mistakes such as incorrect configuration settings or lambda function script errors can be viewed in Cloud Watch.

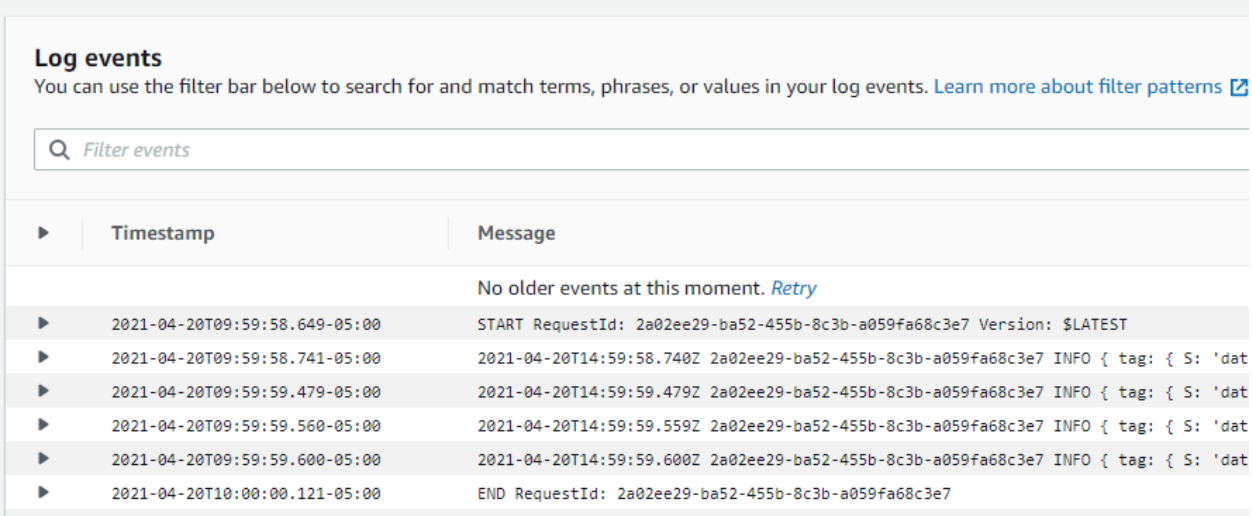

Fig. 23: Cloud Watch Log

5. DynamoDB Table

Lastly, data pushed into AWS can also be seen in the DynamoDB table. Once data is stored it is now available to be used by other AWS resources.

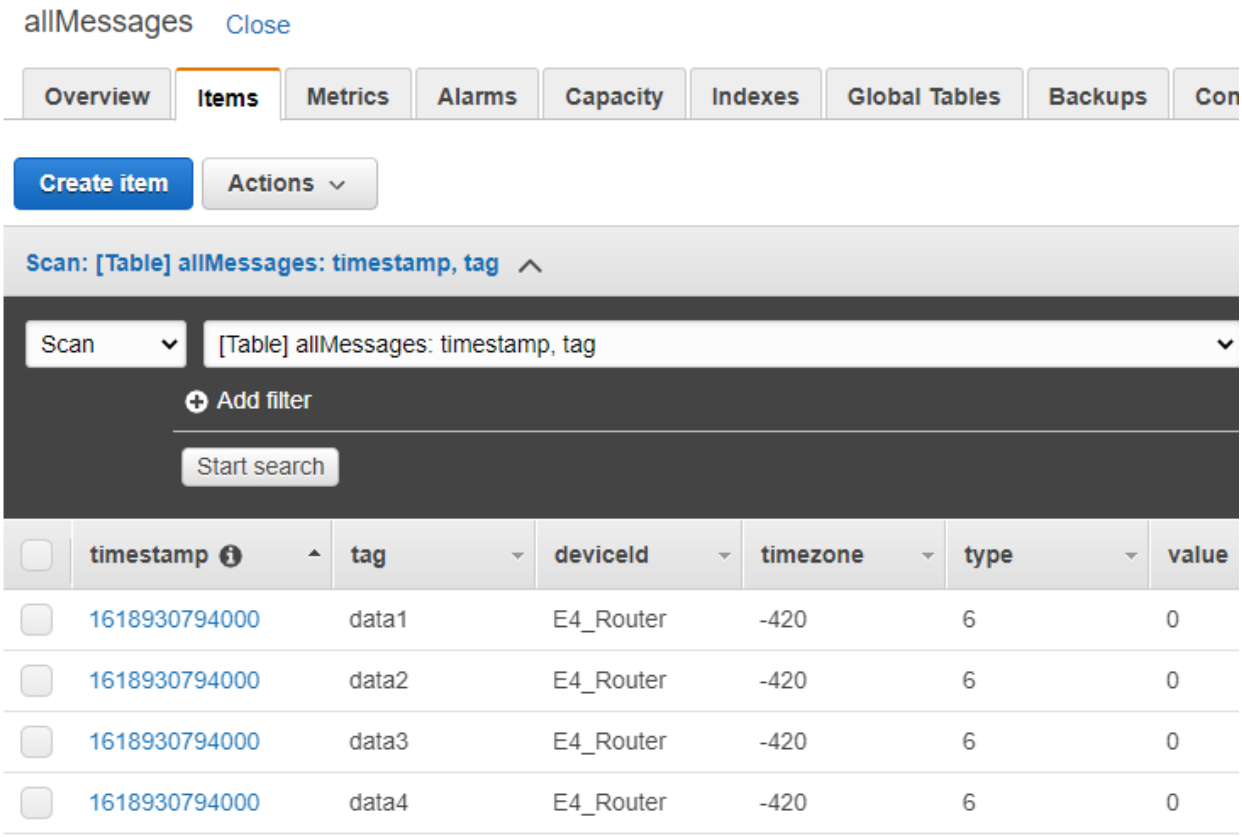

Fig. 24: DynamoDB table

Disclaimer

KEB America, Inc. reserves the right to change/adapt specifications and technical data without prior notification. The safety and warning reference specified in this manual is not exhaustive. Although the manual and the information contained in it is made with care, KEB does not accept responsibility for misprint or other errors or resulting damages. The marks and product names are trademarks or registered trademarks of the respective title owners.

The information contained in the technical documentation, as well as any user-specific advice in verbal or in written form are made to the best of our knowledge and information about the application. However, they are considered for information only without responsibility. This also applies to any violation of industrial property rights of a third-party.

Inspection of our units in view of their suitability for the intended use must be done generally by the user. Inspections are particular necessary, if changes are executed, which serve for the further development or adaption of our products to the applications (hardware, software or download lists). Inspections must be repeated completely, even if only parts of hardware, software or download lists are modified.

Application and use of our units in the target products is outside of our control and therefore lies exclusively in the area of responsibility of the user.

Americas:

KEB America, Inc.
5100 Valley Industrial Blvd South
Shakopee, MN 55379, USA
(+1) 952-224-1400
info@kebamerica.com

Headquarters:

KEB Automation KG
Suedstrasse 38
D - 32683 Barntrup, Germany
(+49) 5263 401-0
info@keb.de