**PLC And AWS Configuration Manual**

**FOR**

**MQTT Application**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| **REvision details** | **REVISION** | **REV. DATE** | **PREP. BY** | **CHECKED** | **APPROVED** |

COPY RIGHT

*All rights reserved by IC Pro OG Technologies Pte. Ltd. Reproduction or issue is not permitted without the written permission of IC Pro OG Technologies Pte. Ltd.*

Contents

[1 Introduction 4](#_Toc111040547)

[1.1 Overview 4](#_Toc111040548)

[1.2 Principle of Operation 5](#_Toc111040549)

[1.3 Components Used 5](#_Toc111040550)

[2 AWS Configuration 6](#_Toc111040551)

[3 Creating PLC Certificate with OpenSSL 11](#_Toc111040552)

[4 PLC Configuration 13](#_Toc111040553)

[4.1 Creating A new project in TIA Portal V16 13](#_Toc111040554)

[4.2 Adding LMQTT Library 16](#_Toc111040555)

[4.3 Integration of LMQTT Function block in the user program 18](#_Toc111040556)

[4.4 Calling Function Blocks in the User Program 21](#_Toc111040557)

[4.5 Adding AWS Certificate in PLC 23](#_Toc111040558)

[4.6 Configuring Router, DNS and Enabling time synchronization via NTP server 26](#_Toc111040559)

[5 Browser application 29](#_Toc111040560)

[5.1 Overview 29](#_Toc111040561)

[5.2 Installation 29](#_Toc111040562)

[5.3 Using SDK with webpack 30](#_Toc111040563)

[5.4 Setting credentials in a web browser 30](#_Toc111040564)

# Introduction

## Overview

Motivation

Digitization has a major impact on the economy and society and is progressing inexorably. The "Internet of Things" (short: IoT) is one of the main drivers of digitization. The term "Internet of Things" is synonymous with one of the biggest current dynamics of change: the increasing networking and automation of devices, machines, and products. The protocol "Message Queue Telemetry Transport" (short: MQTT) is used in the "Internet of Things" as a communication protocol. Its lightweight approach opens new possibilities for automation.

Slim and quick: MQTT

The MQTT is a simple built-in binary publish and subscribe protocol at the TCP/IP level. It is suitable for messaging between low-functionality devices and transmission over unreliable, low-bandwidth, high-latency networks. With these characteristics, MQTT plays an important role for IoT and in M2M communication.

Criteria of MQTT

The MQTT protocol is distinguished by the following criteria:

* Lightweight protocol with low transport overhead
* Minimal need for network bandwidth through push mechanism
* Function for re-connection disconnection
* Re-sending messages after disconnection
* Mechanism for notifying interested parties after an unpredicted disconnection of a client
* Simple use and implementation thanks to a small set of commands
* Quality of Service (QoS level) with different reliability levels for the message delivery
* Optional encryption of messages with SSL/TLS
* Authentication of publishers and subscribers with username and password

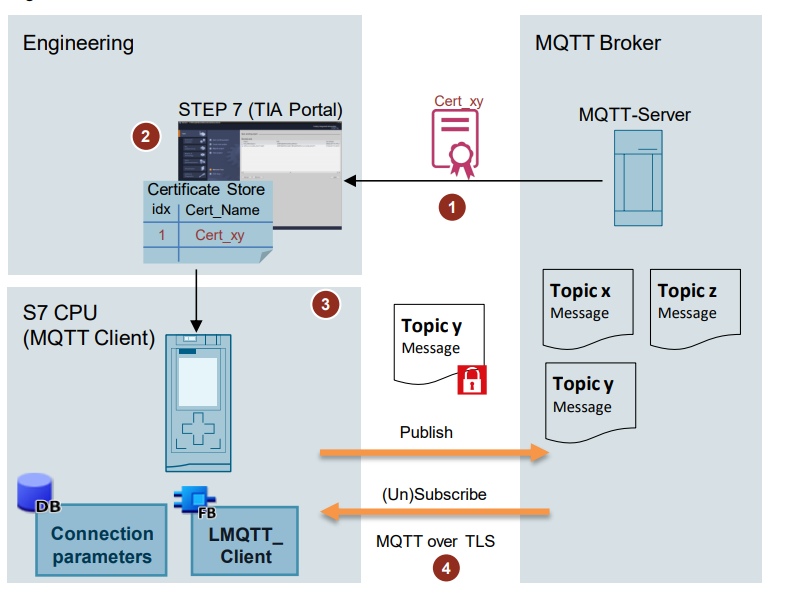
Applicative implementation

To implement the MQTT protocol in a SIMATIC S7 Controller, the "LMQTT" library offers an adequate solution. The "LMQTT" library provides a function block for the SIMATIC S7-1500 and SIMATIC S7-1200. The function block "LMQTT Client" integrates the MQTT Client function and allows you to submit MQTT messages to a broker (Publisher role) and to create subscriptions (Subscriber role). The communication can be secured via a TLS connection

## Principle of Operation

Schematic representation

The following figure shows the most important relationships between the components involved and the steps required for secured MQTT communication (MQTT over TLS)



## Components Used

PLC: Simatic S71200, CPU 1212CDC/DC/DC

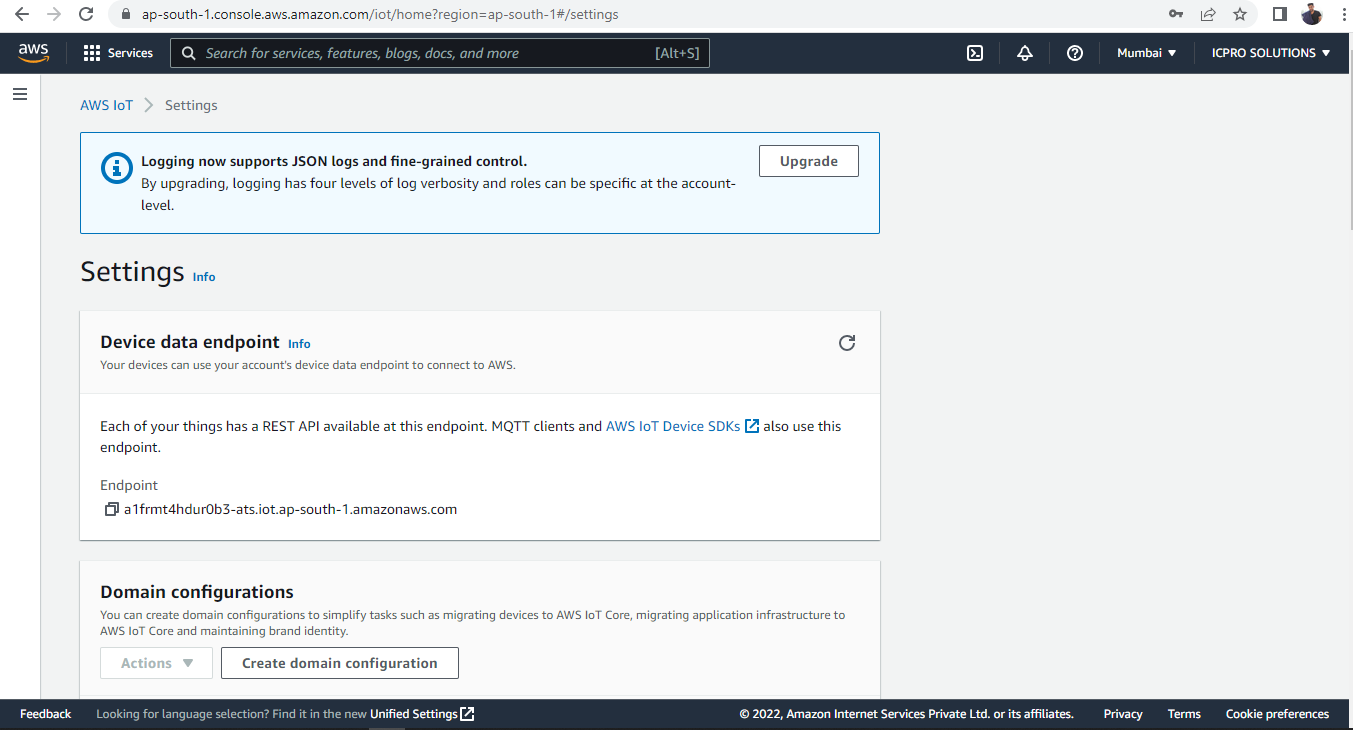
Programming Software: TIA Portal V 16

Cloud: AWS

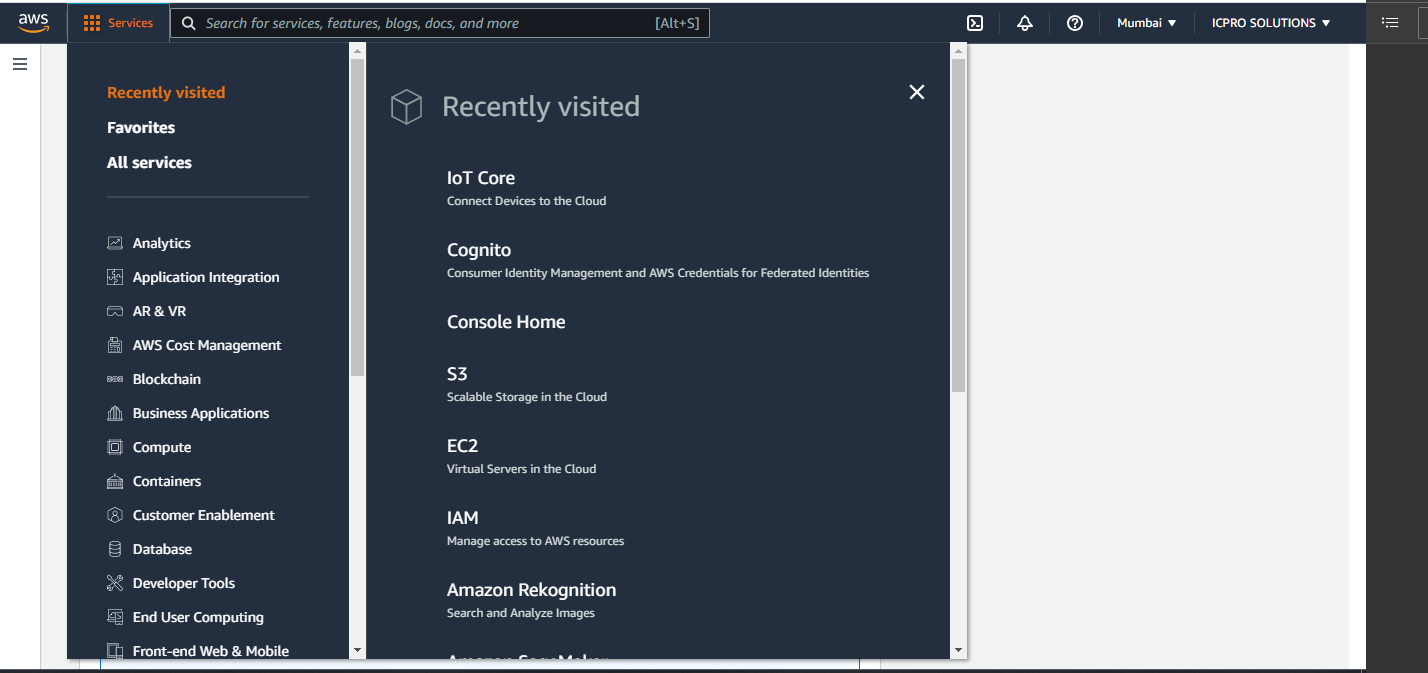
MQTT Broker: AWS

# AWS Configuration

* Log In to Amazon web services
* Click On Services



* Select IOT Core from the drop-down menu



* Under Manage Expand All devices and click on ‘Things’

Graphical user interface, text

Description automatically generated

* Click on ‘Create Things’ from the opened window

Graphical user interface, text, application, email

Description automatically generated

* Select Create single thing and click on Nex

Graphical user interface, text, application, email

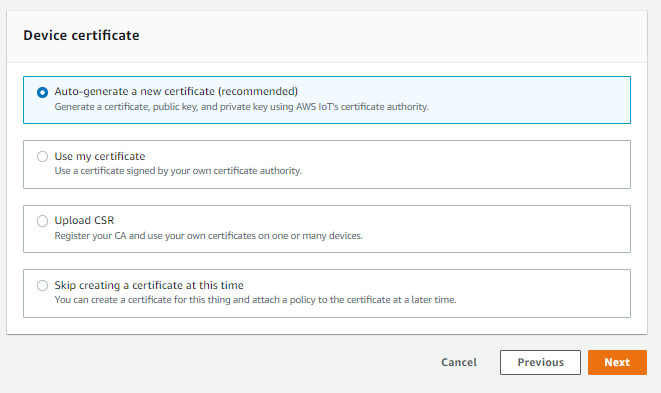
Description automatically generated

* Enter ‘Thing Name’ and Select ‘No shadow’ then click on Next

Graphical user interface, text, application, email

Description automatically generated

* Select ‘Auto generate new certificate’ and click on Next.

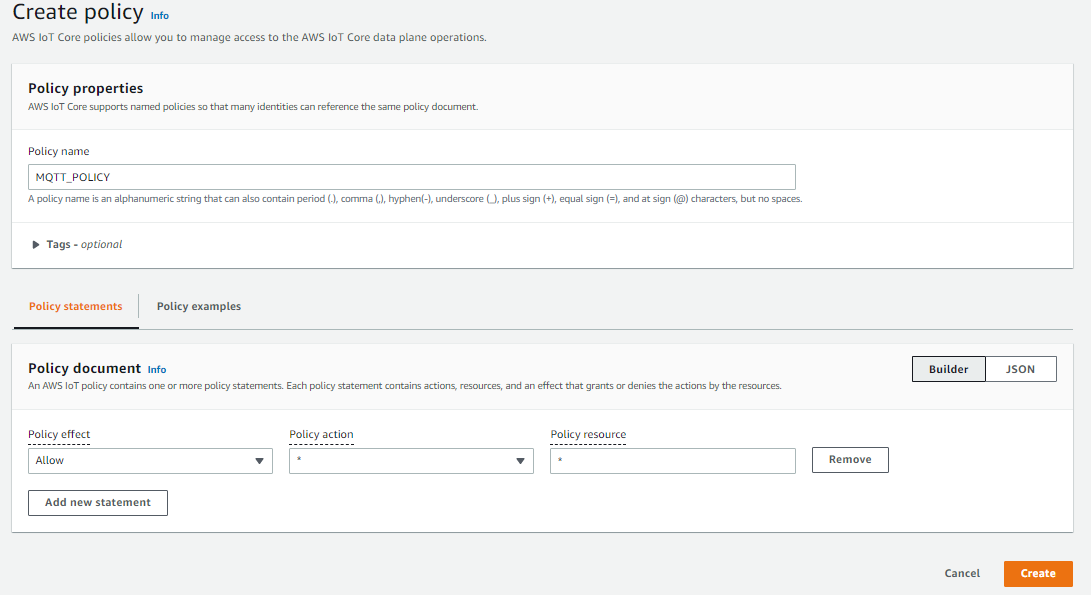


* Click On create policy. A new tab in the browser will be opened.

Graphical user interface, text, application, email

Description automatically generated

* Enter a policy name and under policy document enter ‘\*’ in both policy action and policy resource and click on create.



* Select the previous tab in the browser and select the policy created and click on Create Thing.

Graphical user interface, text, application, email

Description automatically generated

* Download the certificate and keys without fail. This is the only time you can download the key files for the certificate

Graphical user interface, text, application, email

Description automatically generated

* Once all the certificate and keys downloaded click on done.

# Creating PLC Certificate with OpenSSL

* Download Open SSL from the below link

<https://sourceforge.net/projects/openssl/>

* Add the path in the environment variable

Graphical user interface, application

Description automatically generated

* Create New User variable as OPENSSL\_CONF and add the path for openssl.cnf file under variable value

Graphical user interface, text, application, email

Description automatically generated

* Open the folder where the AWS certificate saved
* Create A new text document as Key
* Open The text document and paste downloaded private key and certificate

Text

Description automatically generated with medium confidence



* Open Command prompt as administrator and open the directory where you save the AWS certificate
* Use command cd <certificate path> and press enter

Text

Description automatically generated

* Type command openssl

Text

Description automatically generated

* Enter the Command for creating plc key

pkcs12 -export -in Key.txt -out PLCKey.p12

Enter if it is required to protect the key or avoid it by pressing enter. The PLC Key will be generated in the same path where your certificates saved

Text

Description automatically generated

Text

Description automatically generated

# PLC Configuration

## Creating A new project in TIA Portal V16

* Open TIA Portal V16
* Click on Create new project

Graphical user interface

Description automatically generated

* Enter the project Name, Path and Author and click on Create

Graphical user interface, application

Description automatically generated

* Click on Configure a device

Graphical user interface

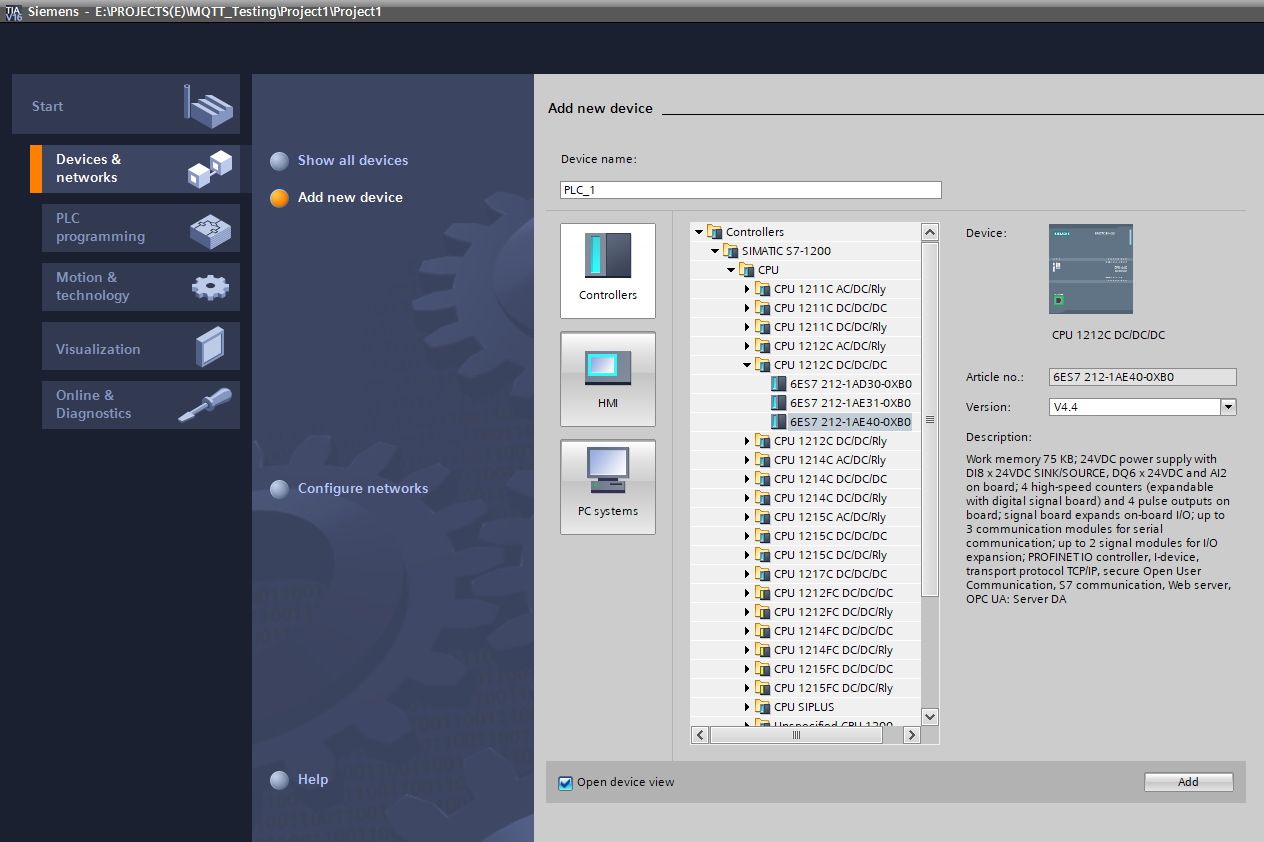
Description automatically generated

* Click on Add new device

Background pattern

Description automatically generated with medium confidence

* Select The controller from the list (For this sample program S71200C DC/DC/DC is used) and click on Add



* Parameterize the ethernet interface of the CPU with an IP address that lies in the same subnet of the network.

Graphical user interface

Description automatically generated

## Adding LMQTT Library

* LMQTT Library can be downloaded from the below link

<https://support.industry.siemens.com/cs/ww/en/view/109780503>

Graphical user interface, text, application

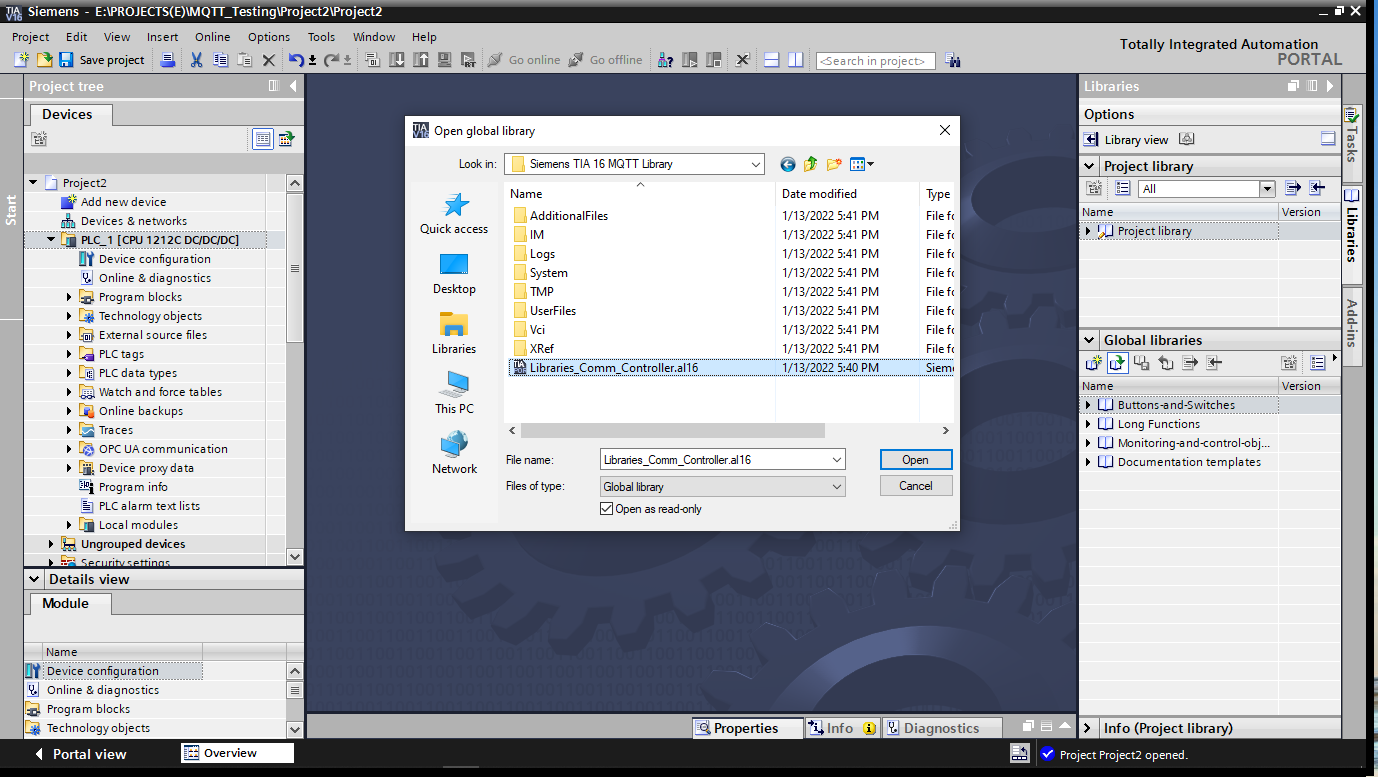
Description automatically generated

* In the TIA portal project, Click On the “Libraries” Task card and open the “Global Libraries” palette

Graphical user interface

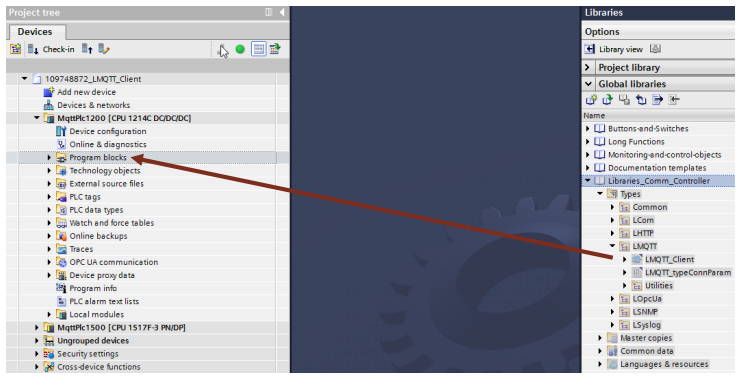
Description automatically generated

* Click on the open Global Library button. Browse the path and select LMQTT Library File Then Click Open



## Integration of LMQTT Function block in the user program

* Insert the function block for your CPU via drag & drop into the folder "Program blocks" of your device.



* The data types used by the FB "LMQTTClient" are automatically inserted into the folder "PLC data types" on your device

Graphical user interface, text, application

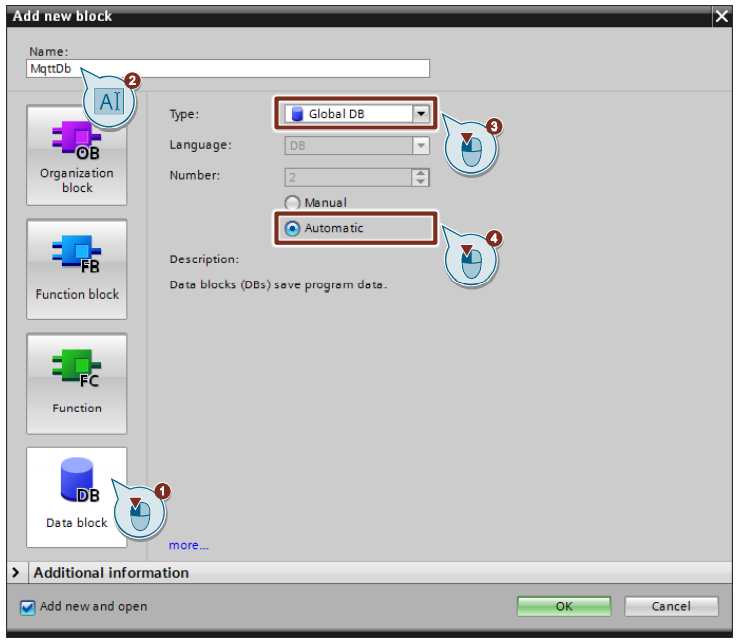
Description automatically generated

* Create a global DB to store
* TCP Connection Parameters
* MQTT Connection Parameters
* Topic and message to be send to the MQTT Broker (Publish)
* Received data, i.e., message and name of the subscribed topic (subscribe)

1. Navigate in the "Project tree" to the device folder of the CPU.

2. Open the "Program blocks" folder and double-click the "Add new block" command. The dialog "Add new block" opens.

* Make the following settings in the below fig and then confirm your entries with the "OK" button.



* Double-click the newly inserted global data block to open it
* The following figure shows the tags in the DB "MqttDb" for switching the inputs and outputs of the FB "MQTT\_Client".

Graphical user interface, table

Description automatically generated with medium confidence

The following figure shows the parameters of the tag "connparams".

Table

Description automatically generated

## Calling Function Blocks in the User Program

* In the "Project tree" open the folder "Program blocks" of your CPU
* Double-click the block "Main [OB1]" to open the corresponding program editor.
* Drag & drop the FB "LMQTT\_Client" from the project navigation to any OB1 network.

Graphical user interface, text, application, email

Description automatically generated

* The dialog "Call options" for generating the instance DB of the FB "LMQTT\_Client" opens automatically.
* Make the following settings and then confirm your entries with the "OK" button.

– Enter the name of the instance DB.

– Enable the "Automatic" radio button for automatic number assignment. The number of the instance DB is assigned by the TIA Portal.

– Click "OK" to confirm the settings.

Graphical user interface, text, application, chat or text message

Description automatically generated

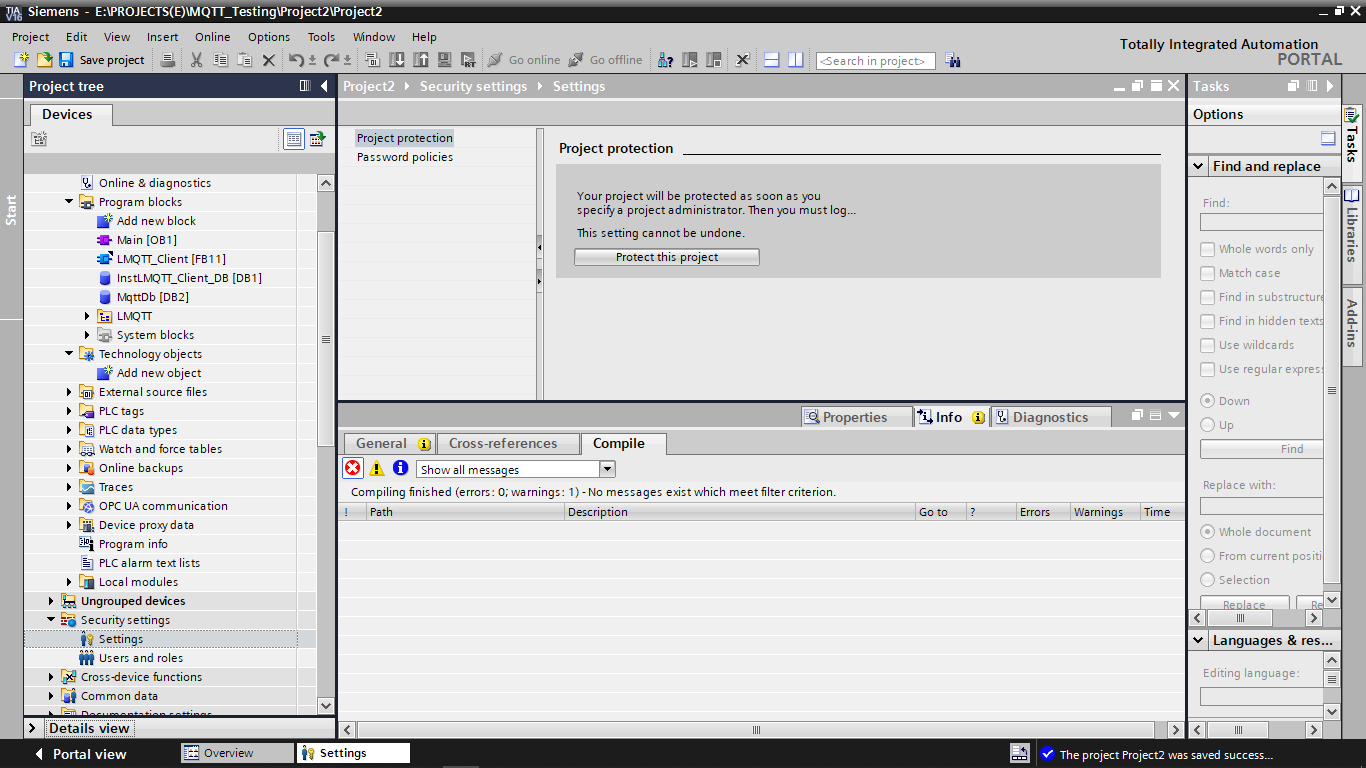
* The following figure shows the linking of the DB “MqttDb”Tags on the FB “MQTT Client”

Timeline

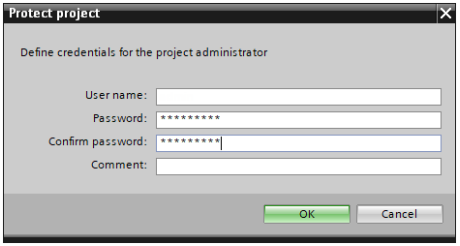
Description automatically generated with medium confidence

## Adding AWS Certificate in PLC

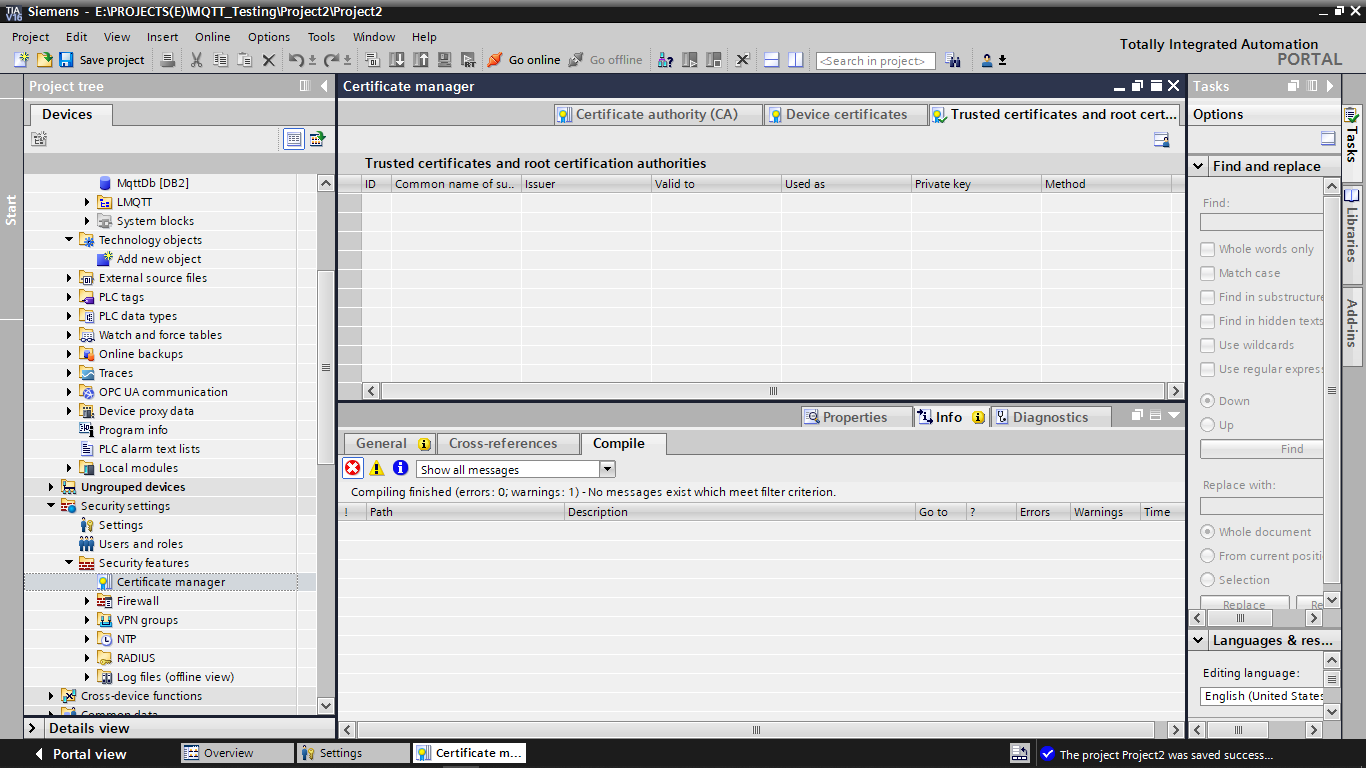
* Under Security settings click on settings. From the opened window click on protect the project



* Provide a username and password the click ok



* You have activated the user administration. You are logged in as a project administrator and can use the security settings. If you have logged in, a line "Certificate manager" appears under the entry "Security settings > Security features"
* Double-click the "Certificate manager" entry in the project navigation under "Security settings > Security features".and click on the “Trusted certificates and core certification authorities" to import the PLC key and Root certificate created earlier.

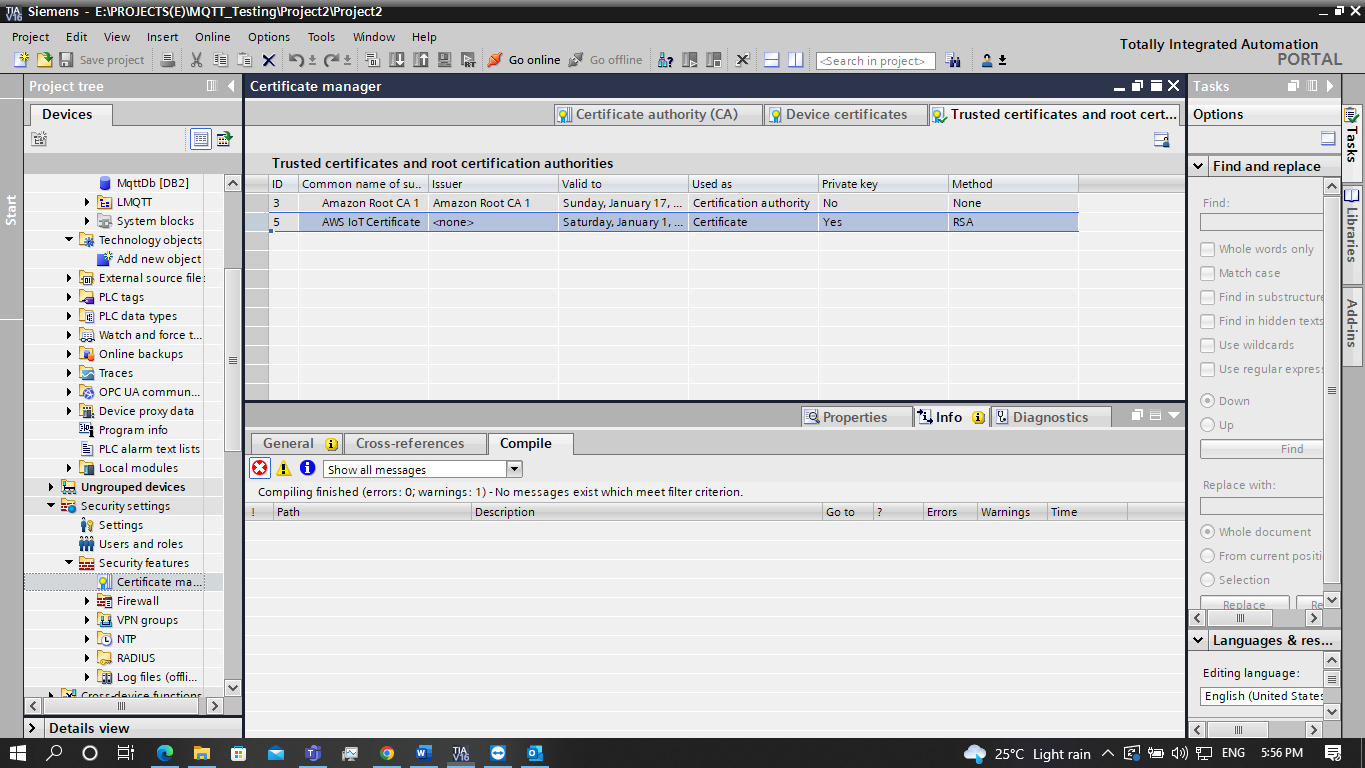


* To open the context menu, right-click in the tab. Click "Import".

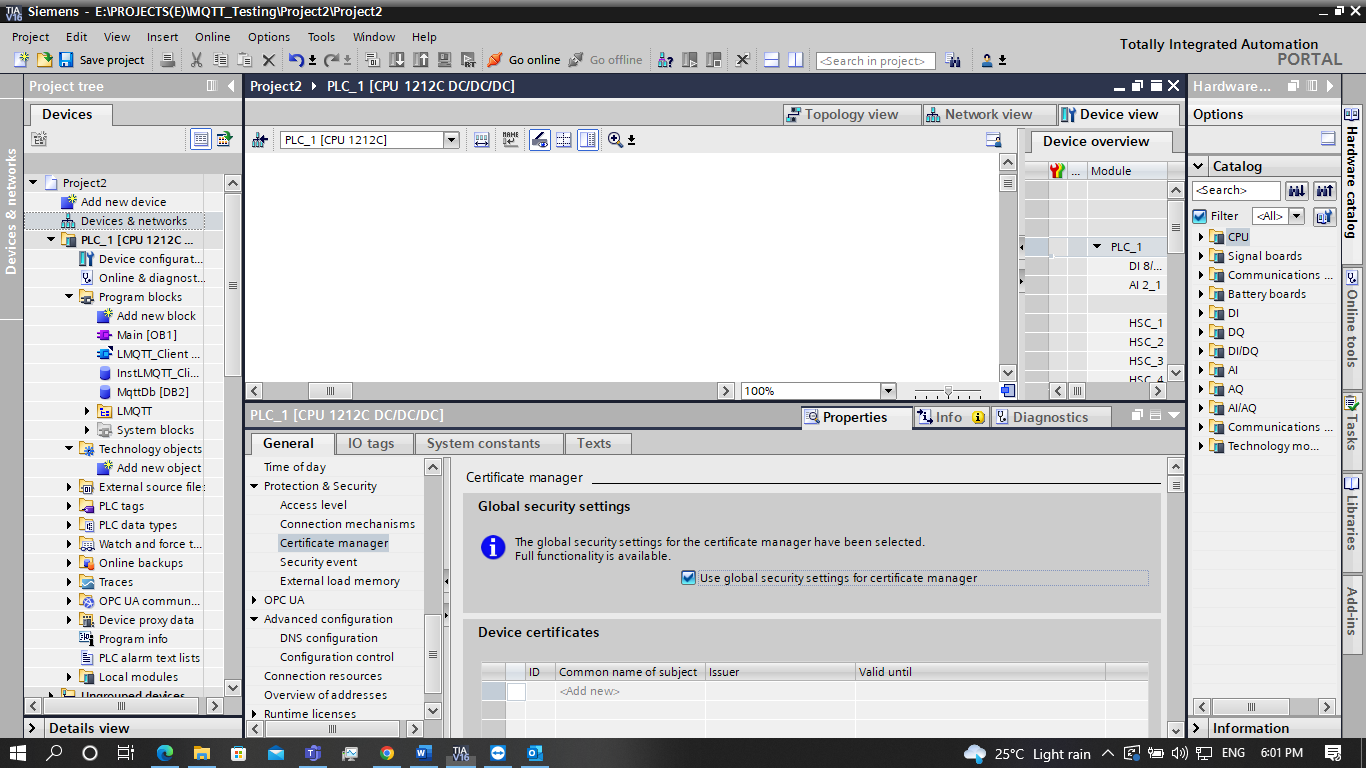
Table

Description automatically generated with medium confidence

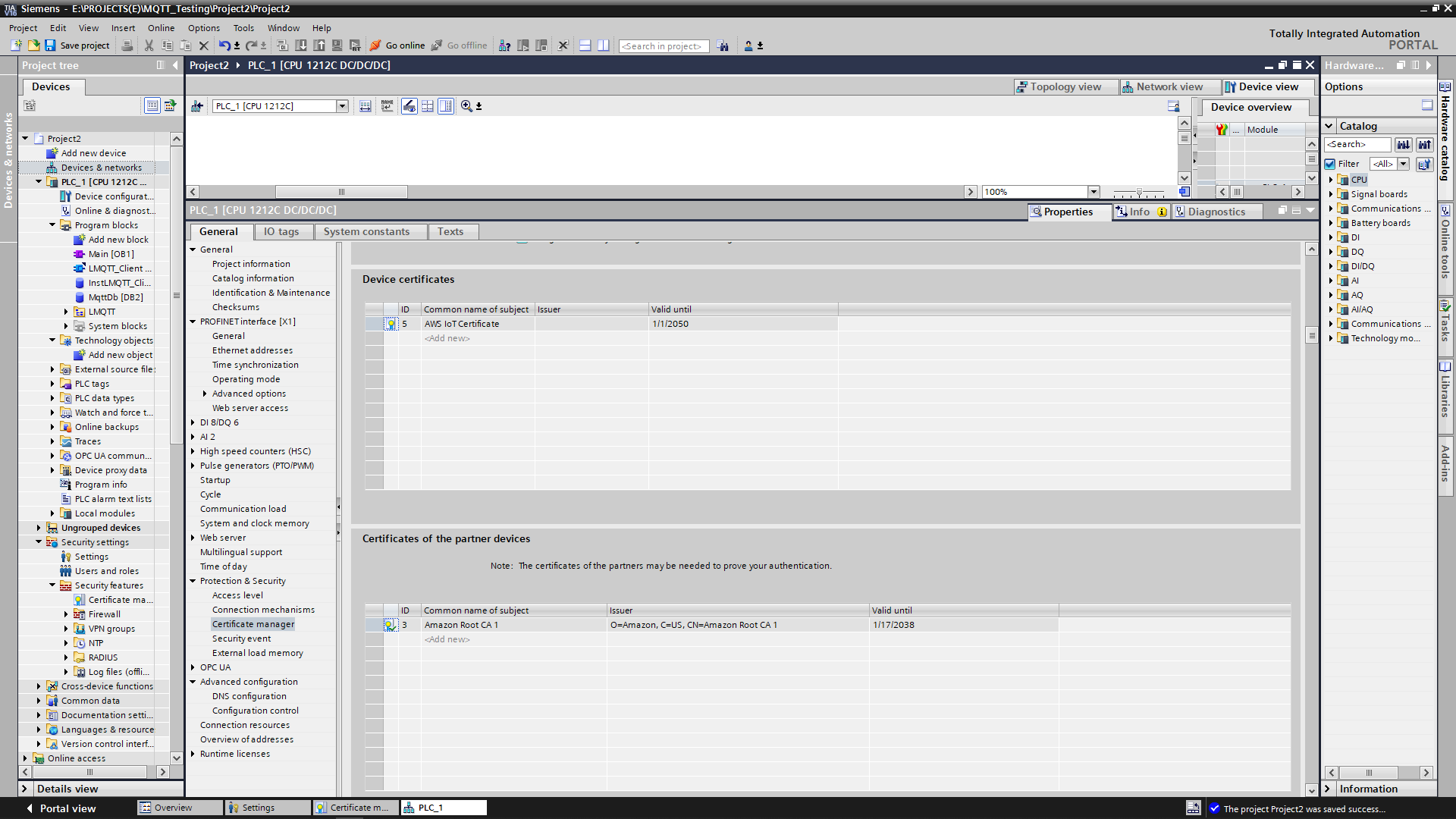
* Import the Amazon root CA1 and PLC Key created earlier and note down the id for ref.



* To authenticate the MQTT broker, you have to load the CA certificate to the CPU. Open Device and Networks and expand Protection & Security. Under Protection & Security select certificate manager and enable “Use global security settings for certificate manager.



* Click on the empty field and add AWS iot Certificate under Device certificate and Amazon Root CA1 Certificate under Certificate of the partner device



## Configuring Router, DNS and Enabling time synchronization via NTP server

* Open Command Prompt as Administrator and Enter the following command for getting DNS and Default gateway address

Ipconfig /all

Text

Description automatically generated with low confidence

* Enable use router and enter the Default gateway address as Router address

Graphical user interface, text, application

Description automatically generated

* Unser Advanced Configuration Select DNS Configuration and enter the DNS address

Graphical user interface, application

Description automatically generated

* To enable NTP server open command prompt as administrator and enter the following command

Net start w32time

Text

Description automatically generated

* To get the name of NTP Server enter the following command in command prompt

w32tm /query /configuration

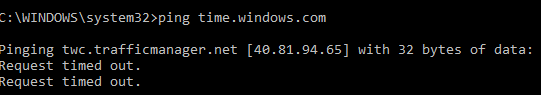
Text

Description automatically generated

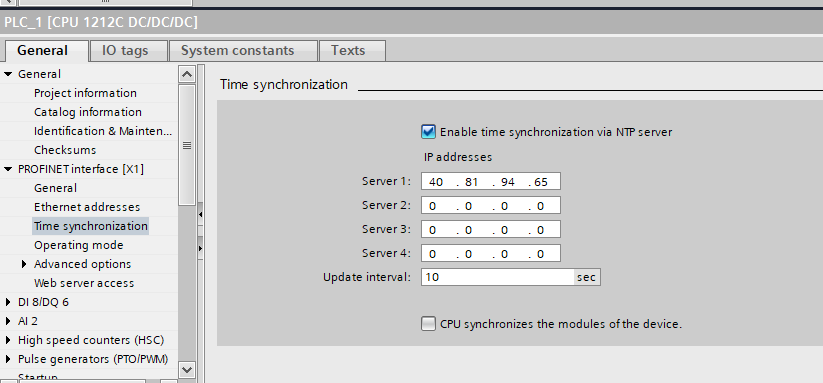
* To get the name of NTP IP address enter the following command in command prompt

ping [Ntp Server name]

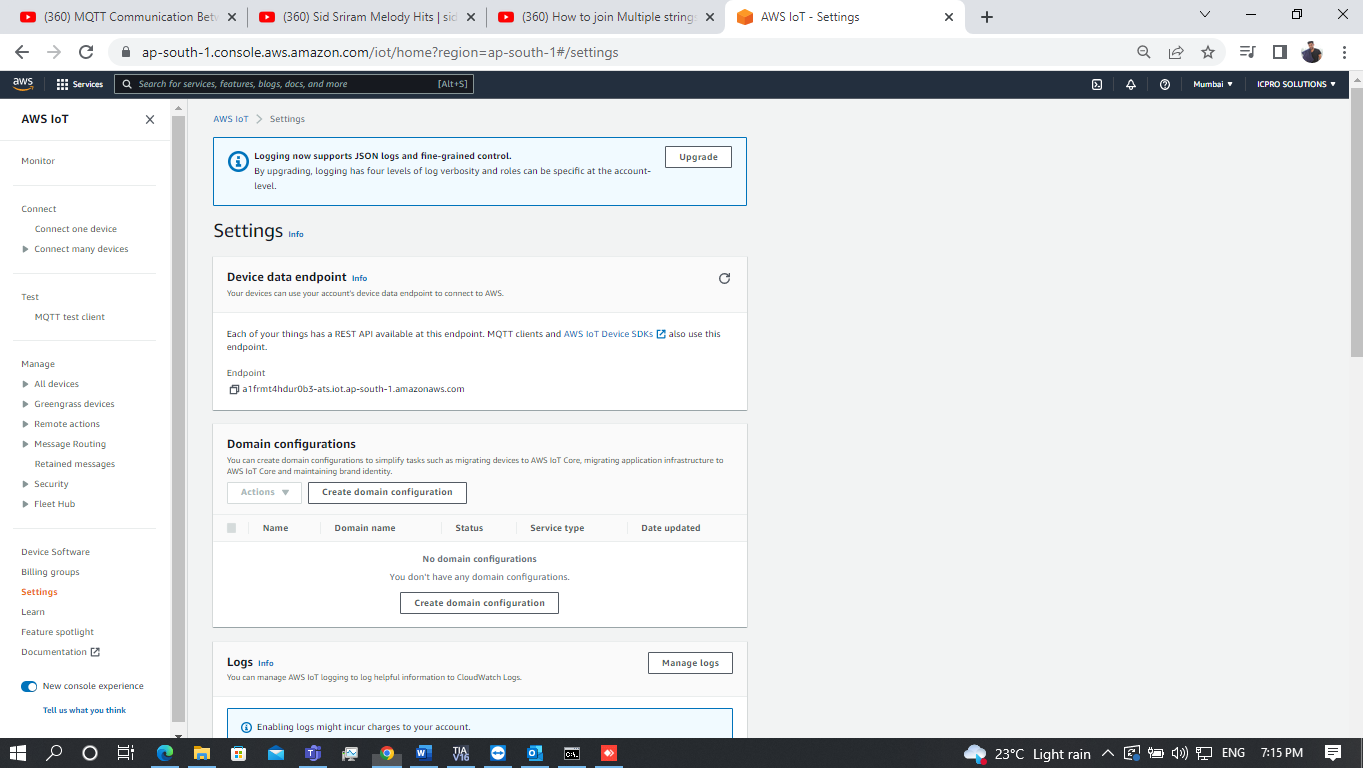
In this example Ntp Server is time.windows.com



* Expand PROFINET interface [X1] and click on Time synchronization. Enable time synchronization via NTP sever and enter the IP Address



* Copy the Endpoint address of AWS and paste the same under the qdn address parameter in connection parameter.add ‘.’ At the end of the qdn address



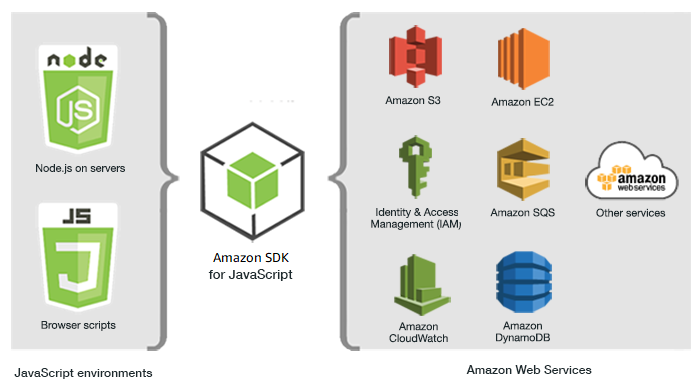
Graphical user interface, text, email

Description automatically generated

* Enter the port no as 8883
* Download the program to PLC

# Browser application

The aws-iot-device-sdk.js package allows developers to write JavaScript applications which access the AWS IoT Platform via [MQTT or MQTT over the Secure WebSocket Protocol](http://docs.aws.amazon.com/iot/latest/developerguide/protocols.html). It can be used in Node.js environments as well as in browser applications.



## Overview

This document provides instructions on how to install and configure the AWS IoT device SDK for JavaScript and includes examples demonstrating use of the SDK APIs.

## Installation

**NOTE:** AWS IoT Node.js SDK will only support Node version 4 or above.

You can check your node version by

*node -v*

Installing from GitHub:

*git clone* [*https://github.com/aws/aws-iot-device-sdk-js.git*](https://github.com/aws/aws-iot-device-sdk-js.git)

*cd aws-iot-device-sdk-js*

*npm install*

## Using SDK with webpack

To work with webpack, you must create a webpack package. You can put your file dependencies in entry.js and output it as bundle.js. An example is provided in the location.

***cd ./examples/browser/mqtt-webpack***

Configure aws configure file

*var awsConfiguration = {*

***poolId: YOUR\_COGNITO\_IDENTITY\_POOL\_ID\_GOES\_HERE, // 'Your Cognito Identity PoolId'***

***host: YOUR\_AWS\_IOT\_ENDPOINT\_GOES\_HERE, // 'Your Aws IoT Endpoint', e.g., 'prefix.iot.us-east-1.amazonaws.com'***

***region: YOUR\_AWS\_REGION\_GOES\_HERE // 'Your Aws Region', e.g., 'us-east-1'***

*};*

***npm install***

***. /node\_modules/.bin/webpack.cmd --config webpack.config.js***

The index.html will load the output file bundle.js and execute functions defined in entry.js.

## Setting credentials in a web browser

**To create an identity pool**

1. Go to the [Amazon Cognito console](https://console.aws.amazon.com/cognito/federated). If prompted, enter your AWS credentials.
2. Choose **Manage Identity Pools**.
3. Choose **Create new identity pool**.
4. Enter a name for your identity pool.
5. To enable unauthenticated identities, select **Enable access to unauthenticated identities** from the **Unauthenticated identities** collapsible section.
6. Choose **Create Pool**.
7. You will be prompted for access to your AWS resources.

Choose **Allow** to create the two default roles associated with your identity pool: one for unauthenticated users and one for authenticated users. These default roles provide your identity pool access to Amazon Cognito Sync. You can modify the roles associated with your identity pool in the IAM console.

1. Make a note of your identity pool Id number. You will use it to set up policies that will allow your app users to access other AWS services, such as Amazon Simple Storage Service or DynamoDB