**SOFTWARE DESIGN SPECIFICATION**

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 Solutions Pte. Ltd.

|  |  |
| --- | --- |
| **Project Name:** | **ED Software** |
| **End Customer Name:** | **IC Pro Solutions Pvt. Ltd** |
| **Document No:** | **ICP-SDS-PR\_EDS\_V0** |
| **Document Title:** | **Software Design Specification** |
| **Prepared by:** | **IC Pro Solutions Pvt. Ltd** |

**REVISION HISTORY**

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

*CHANGE DOCUMENT CONTROL*

*Once IC Pro and Air Liquide Singapore have approved the document, every variation made on the document must be referenced as a change.*

Contents

[1. Prerequisites 4](#_Toc98937636)

[1.1. Supported platforms 4](#_Toc98937637)

[2. Set up Windows Greengrass core device 4](#_Toc98937638)

[2.1. Create an AWS IAM user who has permissions to provision a device 4](#_Toc98937639)

[2.2. Install version 2 of the AWS Command Line Interface (CLI) on the device 5](#_Toc98937640)

[2.3. Create a local user on the Windows machine that will run AWS IoT Greengrass 5](#_Toc98937641)

[2.4. Download and run the AWS IoT Greengrass installer 5](#_Toc98937642)

[2.5. Checking that AWS IoT Greengrass is installed and running. 6](#_Toc98937643)

[2.6. Create and deploy your first AWS IoT Greengrass component 6](#_Toc98937644)

[2.7. Clean Up 8](#_Toc98937645)

[3. Basic OPC\_UA Server simulator code 9](#_Toc98937646)

[4. Basic OPC\_UA Client GUI 10](#_Toc98937647)

[4.1. Install Qt Designer on system 10](#_Toc98937648)

[4.2. Build the GUI of an application’s main window using QT Designer 10](#_Toc98937649)

[4.3. Use Qt Designer’s .ui files in application 10](#_Toc98937650)

[5. Documentation 14](#_Toc98937651)

[5.1. To check if the AWS IoT Greengrass Core software is installed as a system service 14](#_Toc98937652)

[5.2. To run the AWS IoT Greengrass Core software 14](#_Toc98937653)

# Prerequisites

* AWS Account
* AWS CLI V2 or later
* An AWS Identity and Access Management (IAM) user with administrator permissions
* [Python](https://www.python.org/downloads/) 3.5 or later installed for all users on the device and added to the PATH environment variable. On Windows, you must also have the Python Launcher for Windows installed for all users

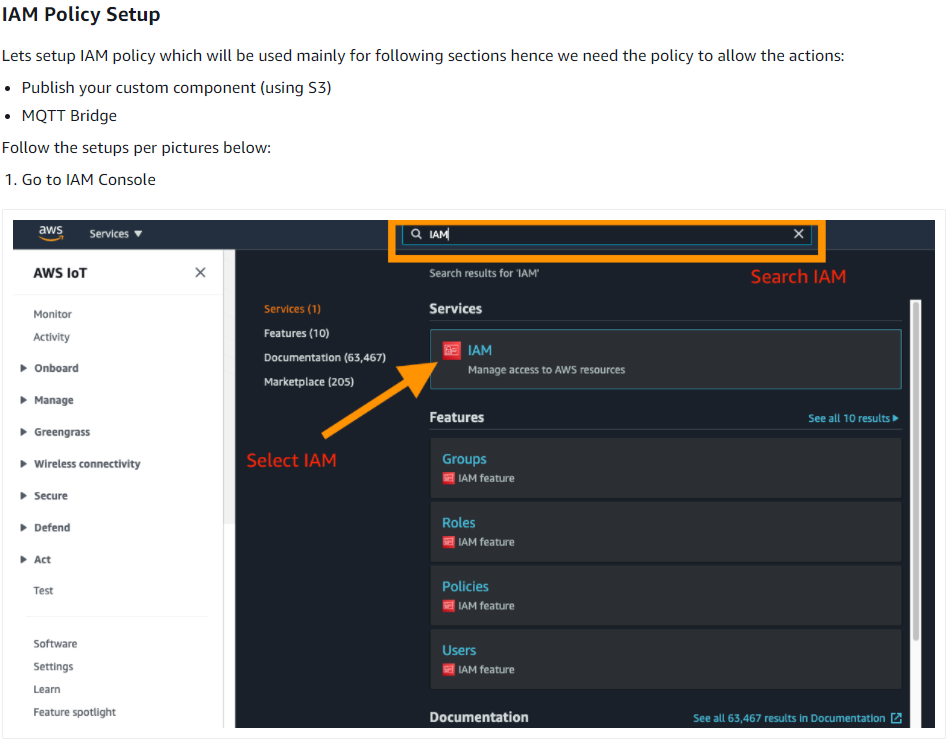
## Supported platforms

* Architectures: X86\_64
* Windows 10 or later Linux
* Windows Server 2019
* Minimum 256 MB disk space available for the AWS IoT Greengrass Core software. This requirement doesn't include components deployed to the core device.
* Minimum 160 MB RAM allocated to the AWS IoT Greengrass Core software. This requirement doesn't include components that run on the core device. For more information, see [Control memory allocation with JVM options](https://docs.aws.amazon.com/greengrass/v2/developerguide/configure-greengrass-core-v2.html#jvm-tuning).
* Java Runtime Environment (JRE) version 8 or greater. To use Java to develop custom components, you must install a Java Development Kit (JDK). We recommend [Amazon Corretto 11](http://aws.amazon.com/corretto/) or [OpenJDK 11](https://openjdk.java.net/).
* Each user that runs component processes must exist in the LocalSystem account, and the user's name and password must be in the Credential Manager instance for the LocalSystem account. You can set up this user when you follow instructions to [install the AWS IoT Greengrass Core software](https://docs.aws.amazon.com/greengrass/v2/developerguide/install-greengrass-core-v2.html).
* Note: To use version 2.5.0 of the [Greengrass nucleus](https://docs.aws.amazon.com/greengrass/v2/developerguide/greengrass-nucleus-component.html), you must use a 64-bit version of the Java Runtime Environment (JRE). Greengrass nucleus version 2.5.1 supports 32-bit and 64-bit JREs.

# Set up Windows Greengrass core device

1. **Create an AWS IAM user who has permissions to provision a device.**
2. **Install version 2 of the AWS Command Line Interface (CLI) on the device**
3. **Create a local user on the Windows machine that will run AWS IoT Greengrass.**
4. **Download and run the AWS IoT Greengrass installer.**
5. **Checking that AWS IoT Greengrass is installed and running.**
6. **Create and deploy your first AWS IoT Greengrass component**.

## Create an AWS IAM user who has permissions to provision a device

1. Open the AWS console and navigate to the [IAM Console](https://console.aws.amazon.com/iam/home).
2. Choose **Users** from left pane.
3. Choose **Add users**.
4. Enter the user name.
5. Select the **Access key in AWS credential type,** as we will only use this IAM user via the CLI.
6. Choose **Next: Permissions**.
7. Choose **Attach existing policies directly**.Graphical user interface, text, application, email

   Description automatically generatedGraphical user interface

   Description automatically generatedGraphical user interface, application

   Description automatically generatedA picture containing table

   Description automatically generated
8. Search and select **IAMFullAccess and Greengrass V2Tokenexchange Role,**. This is used to create the new AWS IoT Greengrass user accounts and roles.Graphical user interface, application

   Description automatically generated
9. Search and select **AWSIoTFullAccess**. This is used to create the device.
10. Search and select **AWSGreengrassFullAccess**. This is used to update the AWS IoT Greengrass service.
11. Choose **Next: Tags**.
12. Choose **Next: Review**.
13. Choose **Create user**.
14. Choose **Download .csv** button to save the login credentials access key and secret access key.
15. Choose **Close**.

## Install version 2 of the AWS Command Line Interface (CLI) on the device

1. Install the AWS CLI on the device. You can follow the [Getting started with the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html#cliv2-windows-install) in the AWS Command Line Interface User Guide. Once installed you will have to reboot the machine.
2. Open command prompt (enter **command prompt** in the windows search bar).
3. Enter “aws **configure**” into the command prompt windows and press the enter key.
4. Enter the **AWS AccessKeyID**
5. Enter the **AWS SecretAccesskey**
6. Enter the **Default region**
7. Enter the default output format type **json**Graphical user interface, text, application, email

   Description automatically generated.

## Create a local user on the Windows machine that will run AWS IoT Greengrass

1. Download PSExec from the [Sysinternals downloads page](https://docs.microsoft.com/en-us/sysinternals/downloads/psexec) and extract the zip file.
2. Copy **psexec.exe to C:\Windows\System32**.
3. Open Command Prompt as Administrator. (Enter **command prompt** in the windows search bar and choose **Run as administrator**.)
4. Choose Yes if prompted by the UAC pop up.
5. Run the following command to create a new user.

net user /add <username> <password>

1. Now run the following command to store the password in the system account so that the AWS IoT Greengrass can access it. Replace <password> with the password you used in the previous step.
2. cd C:\Windows\System32
3. psexec -s cmd /c cmdkey /generic:ggc\_user /user:ggc\_user /pass:<password>

## Download and run the AWS IoT Greengrass installer

1. Login to the AWS console and navigate to [AWS IoT Core](https://console.aws.amazon.com/iot/).
2. Expand **Greengrass** from the left menu and choose **Getting started**.
3. Choose **Set up one core device**.Graphical user interface, text, application

   Description automatically generated
4. For core device name and thing group you may use the default.Graphical user interface, text, application, email

   Description automatically generated
5. Under Step 3: Install the Greengrass Core software, select **Windows**.Graphical user interface, text, application, email

   Description automatically generated
6. Choose **Download.zip or run the following command in command prompt**

**powershell.exe -command "& {Invoke-WebRequest -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip -OutFile .\greengrass-nucleus-latest.zip; Expand-Archive -LiteralPath .\greengrass-nucleus-latest.zip -DestinationPath .\GreengrassCore}"**

1. Unzip the AWS IoT Greengrass v2 files.
2. Open Command Prompt as Administrator. (Enter command prompt in the windows search bar and choose Run as administrator.)
3. Navigate to the folder where you have unzipped the files.
4. Open command prompt and enter the following command and change the appropriate **–aws-region** and **default-user** and press **enter**

**java "-Droot=C:\greengrass\v2" "-Dlog.store=FILE" -jar .\GreengrassCore\lib\Greengrass.jar --aws-region us-east-1 --thing-name GreengrassQuickStartCore-17fb5abe6b3 --thing-group-name GreengrassQuickStartGroup --component-default-user ggc\_user --provision true --setup-system-service true --deploy-dev-tools true**

## Checking that AWS IoT Greengrass is installed and running.

1. Navigate to the [AWS IoT Core](https://console.aws.amazon.com/iot/home) console.
2. Select **Greengrass**.
3. Choose **Core devices**.
4. The device you created in step 6 will be listed under **Greengrass core devices**.
5. Check that AWS IoT Greengrass is installed and running as a system service by opening **Services** (enter **services** in the Windows search bar).
6. Review the list of services for **greengrass** and the status is **running**. If not, repeat the installation steps in step 6 and check for errors. You can also review [Troubleshooting AWS IoT Greengrass V2](https://docs.aws.amazon.com/greengrass/v2/developerguide/troubleshooting.html) in the Developer Guide.

## Create and deploy your first AWS IoT Greengrass component

1. Create a file called **Python file** with the following contents
2. We will store this file in Amazon S3 under a folder structure that can be used to manage the versions.
3. Navigate to [Amazon S3 Console](https://s3.console.aws.amazon.com/s3/home).
4. Choose **Create Bucket**.
5. Give your bucket a globally unique name (greengrasscomponents + a random string such as the date/time of creation).
6. Choose the Region that your AWS IoT Greengrass device was installed.
7. Choose **Create bucket**.
8. When you return to the previous screen select the bucket name that you just created.
9. Choose **Create folder**.
10. For Folder name, enter **artifacts**.
11. Navigate into that folder and create another folder called **com.example.software**.
12. Navigate into that folder and create another folder called **1.0.0** this will be used to hold our first version of the code.
13. You should now have a folder structure that resembles **greengrasscomponents\_<unique name>/artifacts/com.example.software/1.0.0/**
14. Choose **Upload** and select Hello.py file that you created earlier into the 1.0.0 folder.
15. Now select the link **software.bat** file in this folder.
16. Choose the **S3 URI** to copy that link. We will need that in the recipe file.

Now we need to make sure the AWS IoT Greengrass role has access to read the files from the Amazon S3 bucket before we can try to deploy the component.

1. Navigate to the [AWS IoT Core](https://console.aws.amazon.com/iot/home) console.
2. Select **Secure**.
3. Select **Role Aliases**.
4. The role alias that was created during the AWS IoT Greengrass installer process will be listed.
5. Choose Edit IAM Role
6. The link should open the IAM console for that role.
7. Choose the Permissions tab. A GreengrassV2TokenExchangeRoleAccess policy is listed. You may have to expand the list.
8. Choose Edit Policy
9. Choose + Add additional permissions.
10. Choose the service Amazon S3 by expanding the Choose a Service and searching for S3.
11. Next expand the Read section under the Access level heading and select the following check boxes GetObject and GetBucketLocation.
12. Expand the Resources section using the arrow.
13. Choose Add ARN next to bucket. A window will open that will prompt you to enter your bucket name in my case this is greengrasscomponents\_<unique name>.
14. Select Add.
15. Choose Add ARN next to object. A window will open that will prompt you to enter the bucket name and object name. Enter your bucket name (in my case this is greengrasscomponents\_<unique name>).
16. Select the Any box next to the Object name entry box to allow the policy to get all objects in this bucket.
17. Choose **Review policy**.
18. Choose **Save Changes**.

Now we are ready to create our first component to deploy and run on our Windows AWS IoT Greengrass device.

1. Navigate to the [AWS IoT Core](https://console.aws.amazon.com/iot/home) console.
2. Choose **Components**.
3. Choose **Create component**.
4. Enter the recipe below

{

"RecipeFormatVersion": "2020-01-25",

"ComponentName": "com.example.software",

"ComponentVersion": "1.0.0",

"ComponentDescription": "HelloWorld",

"ComponentPublisher": "Me",

"ComponentConfiguration": {

"DefaultConfiguration": { }

},

"Manifests": [

{

"Name": "windows",

"Platform": {

"os": "windows"

},

"Lifecycle": {

"Run": "{artifacts:path}/ Hello.py "

},

"Artifacts": [

{

"Uri": "s3://<bucket\_name>/artifacts/com.example.software/1.0.0/Hello.py"

}

]

}

]}

1. Choose Create component.
2. Choose Deploy.
3. Select your Deployment group for your Windows device.
4. Select the deploy button. You will return to the deployment screen where you can select the IoT Job which will show you the current status of the deployment. Wait until the deployment status is marked as successful before moving to the next step.
5. On your windows device navigate to the AWS IoT Greengrass work folder (C:\greengrass\v2\work\com.example.software) using windows explorer. The batch file has created a hardware.txt file that will contain the results from the script.

## Clean Up

With AWS IoT Greengrass you only pay for what you use. You are charged per device connected so by disconnecting the device, you will no longer be charged. You can find the current pricing on the product page under [AWS IoT Greengrass pricing](https://aws.amazon.com/greengrass/pricing/).

To delete the device from your AWS IoT account, follow these steps:

1. Navigate to the AWS IoT Core console.
2. Select **Manage** then select **Things**.
3. Select the checkbox next to your device and choose **Delete**.
4. Confirm the name of the device in the prompt and choose **Delete**.

You will be charged for the storage of the artifacts within the Amazon S3 bucket. To delete the objects in the Amazon S3 bucket and the bucket, follow the steps in the Amazon Simple Storage Service (S3) User Guide under [Deleting a bucket](https://docs.aws.amazon.com/AmazonS3/latest/userguide/delete-bucket.html).

# Basic OPC\_UA Server simulator code

# Import following modules and class

import datetime

from opcua import Server

from random import randint

from datetime import date

import time

server = Server()

url = "opc.tcp://127.0.0.1:64579"

# Setting server url

server.set\_endpoint(url)

# Assigning server name

name = "OPCUA\_SIMULATION\_SERVER"

addspace = server.register\_namespace(name)

node = server.get\_objects\_node()

Param =node.add\_object(addspace,"Parameters")

# Tags = node.add\_object(addspace,"Tags")

Temp=Param.add\_variable(addspace,"Temperature",0)

Press=Param.add\_variable(addspace,"Pressure",0)

Humidity =Param.add\_variable(addspace,"Humidity",0)

Time=Param.add\_variable(addspace,"Time",0)

Humidity.set\_writable()

Temp.set\_writable()

Press.set\_writable()

Time.set\_writable()

server.start()

print("Server started at {}".format(url))

# Generating node raandom values

while True:

    Temperature = randint(10,50)

    Pressure = randint(200,999)

    TIME =datetime.datetime.now()

    Humid = randint(30,99)

    print("Temperature : ",Temperature )

    print("Pressure : ", Pressure)

    print("Humidity :" ,Humid)

    print("Time : ",TIME )

    Temp.set\_value(Temperature)

    Press.set\_value(Pressure)

    Time.set\_value(TIME)

    Humidity.set\_value(Humid)

    time.sleep(2)

# Basic OPC\_UA Client GUI

## Install Qt Designer on system

## Build the GUI of an application’s main window using QT Designer

## Use Qt Designer’s .ui files in application

Graphical user interface, text, application, email

Description automatically generated

from logging import error

import sys

import os.path

from PyQt5.QtWidgets import \*

from PyQt5.QtCore import\*

from PyQt5.QtGui import \*

from PyQt5 import uic

from opcua import Client

import time as t

from awscrt import io,mqtt

from awsiot import mqtt\_connection\_builder

import json

from opcua.ua.uaprotocol\_hand import Message

class AppDemo(QMainWindow):

    Node\_id = []

# Define ENDPOINT, CLIENT\_ID, PATH\_TO\_CERTIFICATE, PATH\_TO\_PRIVATE\_KEY, PATH\_TO\_AMAZON\_ROOT\_CA\_1, MESSAGE, TOPIC, and RANGE

    ENDPOINT = " ap-south-1.amazonaws.com"

    CLIENT\_ID = "1"

    PATH\_TO\_CERTIFICATE = "certificate.pem.crt"

    PATH\_TO\_PRIVATE\_KEY = "private.pem.key"

    PATH\_TO\_AMAZON\_ROOT\_CA\_1 = "AmazonRootCA1.crt"

    MESSAGE = "Test Message"

    RANGE = 1

    TOPIC = "test/testing"

    # Spin up resources

    event\_loop\_group = io.EventLoopGroup(1)

    host\_resolver = io.DefaultHostResolver(event\_loop\_group)

    client\_bootstrap = io.ClientBootstrap(event\_loop\_group, host\_resolver)

    mqtt\_connection = mqtt\_connection\_builder.mtls\_from\_path(

                endpoint=ENDPOINT,

                cert\_filepath=PATH\_TO\_CERTIFICATE,

                pri\_key\_filepath=PATH\_TO\_PRIVATE\_KEY,

                client\_bootstrap=client\_bootstrap,

                ca\_filepath=PATH\_TO\_AMAZON\_ROOT\_CA\_1,

                client\_id=CLIENT\_ID,

                clean\_session=False,

                keep\_alive\_secs=6

                )

    print("Connecting to {} with client ID '{}'...".format(

            ENDPOINT, CLIENT\_ID))

    # Make the connect() call

    connect\_future = mqtt\_connection.connect()

    # Future.result() waits until a result is available

    connect\_future.result()

    # Publish message to server desired number of times.

    print('Test Publish')

    mqtt\_connection.publish(topic=TOPIC, payload=json.dumps(MESSAGE), qos=mqtt.QoS.AT\_LEAST\_ONCE)

    print("Published: '" + json.dumps(MESSAGE) + "' to the topic: " + "'test/testing'")

    def \_\_init\_\_(self,parent =None):

        super().\_\_init\_\_(parent)

        # Load the ui file

        uic.loadUi('OPC\_UA\_Client.ui',self)

        CURRENT\_DIRECTORY = os.path.dirname(os.path.realpath(\_\_file\_\_))

        filename = os.path.join(CURRENT\_DIRECTORY, "img/icpro.png")

        self.label.setPixmap(QPixmap(filename))

        # make QTimer

        self.qTimer = QTimer()

        # set interval to 1 s

        self.qTimer.setInterval(2000) # 1000 ms = 1 s

        # connect with OPC Server

        self.ConnectButton.clicked.connect(self.connect)

    def connect(self):

        url = self.Connectionstring.text()

        parameter\_id =""

        client = Client(url)

        try :

            client.connect()

            self.statusbar.setStyleSheet("color:green;")

            self.statusbar.showMessage(url + " :Connected successfully..... ")

            server\_node =client.get\_server\_node()

            objects = client.get\_objects\_node()

            objects\_nodes =objects.get\_children()

            for obj\_id in objects\_nodes:

                if(obj\_id!= server\_node):

                    parameter\_id =obj\_id                 #Fetching the parameter id

            var = client.get\_node(parameter\_id)

            tag\_id = var.get\_variables()

            list =[]

            for id in tag\_id:

                Parameters =client.get\_node(id)

                tag\_name =Parameters.get\_display\_name()

                list.append(tag\_name.Text)

            self.Paralist.addItems(list)

            AppDemo.Node\_id = tag\_id

            client.disconnect()

            # Connect to aws

            #self.connect\_IOT

            # connect timeout signal to signal handler

            self.qTimer.timeout.connect(self.get\_value)

            # start timer

            self.qTimer.start()

        except :

            # clean up, disconnect completely from server

            self.statusbar.setStyleSheet("color:red;")

            self.statusbar.showMessage("Could not connect to server: Communication Error")

    def connect\_IOT(self):

        # connection IOT

        self.statusbar.showMessage("Connecting to {} with client ID '{}'...".format(AppDemo.ENDPOINT,AppDemo.CLIENT\_ID))

        # Make the connect() call

        connect\_future = AppDemo.mqtt\_connection.connect()

        # Future.result() waits until a result is available

        connect\_future.result()

        self.statusbar.showMessage("Successfully connected {} with client ID '{}'...".format(AppDemo.ENDPOINT,AppDemo.CLIENT\_ID))

    def get\_value(self):

        url = self.Connectionstring.text()

        client = Client(url)

        try:

            client.connect()

            nrows = self.tableWidget.rowCount()

            for row in range(0,nrows):

                item = self.tableWidget.item(row, 0)

                item\_text = item.text()

                for index in AppDemo.Node\_id:

                    Parameters =client.get\_node(index)

                    tag\_name = Parameters.get\_display\_name()

                    value = Parameters.get\_value()

                    if item\_text == tag\_name.Text:

                        self.tableWidget.setItem(row,1 ,QTableWidgetItem(str(index)))

                        self.tableWidget.setItem(row,2 ,QTableWidgetItem(str(value)))

                        data ="{} {} {}".format("TAG :"+tag\_name.Text, ", ID :"+ str(index), ", Value :" +str(value))

                        self.write\_log(data)

                        self.publish(data)

                self.statusbar.setStyleSheet("color:black;")

                self.statusbar.showMessage("Retrieving And Publishing  ......")

        except:

            self.statusbar.setStyleSheet("color:red;")

            self.statusbar.showMessage("Could not connect to server: Communication Error")

       #Initialize The App

    def publish(self,message):

        AppDemo.mqtt\_connection.publish(topic=AppDemo.TOPIC, payload=json.dumps(message), qos=mqtt.QoS.AT\_LEAST\_ONCE)

# Writinginto txt file

    def write\_log( self ,message):

        with open('log.txt', 'a') as f:

            f.write(message)

            f.write('\n')

if \_\_name\_\_ == '\_\_main\_\_':

    app =QApplication(sys.argv)

    app.setStyle("fusion")

    demo = AppDemo()

    demo.resize(1000,600)

    demo.show()

    try:

        sys.exit(app.exec\_())

    except SystemExit:

        print("Application Closed")

# Documentation

<https://github.com/FreeOpcUa>

<https://python-opcua.readthedocs.io/en/latest/>

<https://www.tutorialspoint.com/pyqt5/pyqt5_introduction.htm>

<https://realpython.com/qt-designer-python/>

<https://ap-south-1.console.aws.amazon.com/iot/home?region=ap-south-1#/greengrassIntro>

[https://aws.amazon.com/blogs/iot/aws-iot-greengrass-now-supports-the-windows-operating- system/](https://aws.amazon.com/blogs/iot/aws-iot-greengrass-now-supports-the-windows-operating-%20%20%20system/)

<https://ap-south-1.console.aws.amazon.com/iot/home?region=ap-south-1#/greengrass/v2/cores/create>

## To check if the AWS IoT Greengrass Core software is installed as a system service

sc query greengrass

## To run the AWS IoT Greengrass Core software

sc start greengrass