TI Designs: TIDC-01002

Sub-1 GHz Embedded Sensor to Cloud Industrial Internet of Things (IoT) Gateway Reference Design

Description

The TIDC-01002 design demonstrates how to connect sensors to the cloud over a long-range, Sub-1 GHz wireless network, which may be used in industrial settings, such as building control and asset tracking. This TI Design is powered through a TI SimpleLink™ CC3220 processor and SimpleLink ultra-low-power (ULP), Sub-1 GHz CC13x0 and CC13x2 devices. The reference design pre-integrates the TI 15.4-Stack part of the SimpleLink CC13xx software development kits (SDK) and SimpleLink CC3220 SDK, which are part of the TI SimpleLink MCU platform, providing a unified software experience across TI low-power, wired, and wireless MCUs.

Resources

TIDC-01002	Design Folder
TIDC-01002	Repository Page
CC1352	Product Folder
CC1310	Product Folder
CC1350	Product Folder
CC3220	Product Folder

TI E2E™ Community

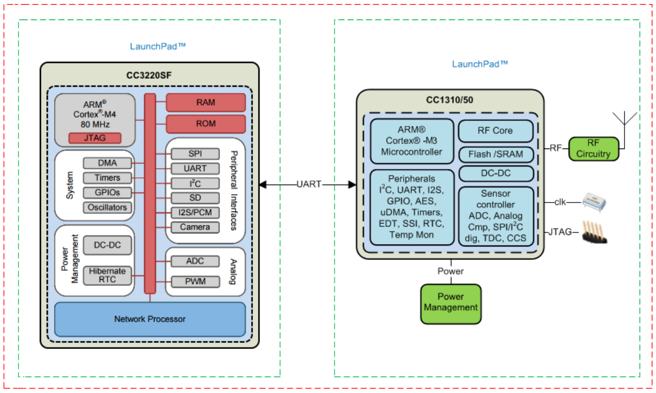
Ask Our E2E Experts

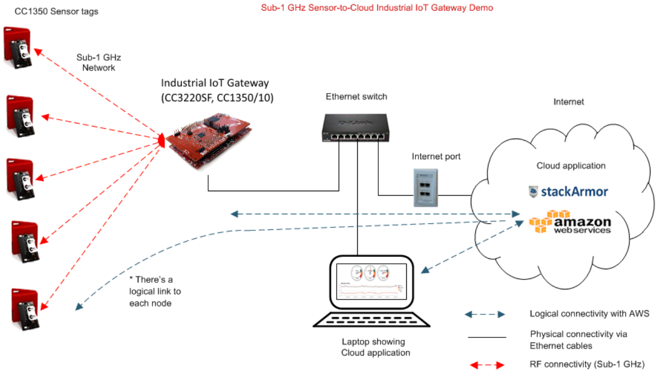
Features

- Large Network-to-Cloud Connectivity Enabling Long Range, Up to 1 km (Line of Sight)
- Facilitates Designer's System Compliance With IEEE 802.15.4e/g by Using TI 15.4-Stack
- Based on TI Tested Hardware Designs Enabling Quick Time to Market With Out-of-the-Box, Ready-to-Use Demonstration Software
- Implementation Based on Portable Operating System Interface (POSIX), Allows for Easy Portability Across TI Internet Connected Microcontrollers (MCUs)
- Supports Star Networks
- ULP Sensor Nodes

Applications

- Building Security Gateway
- Door and Window Sensor Networks
- HVAC Gateway
- Asset Management and Tracking




CC1350 Sensor tags
Sub-1 GHz Sensor-to-Cloud Industrial IoT Gateway Demo
Sub-1 GHz Network
Industrial IoT Gateway (CC3220SF, CC1350/10)
Ethernet switch
Internet
Internet port
Cloud application
stackArmor
amazon web services
* There's a logical link to each node
Laptop showing Cloud application
Logical connectivity with AWS
Physical connectivity via Ethernet cables
RF connectivity (Sub-1 GHz)

An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 System Description

The TIDC-01002 provides a reference for creating an industrial, Internet of things (IoT) gateway that is capable of connecting a network of wireless sensors to an enterprise cloud provider. In this reference design, a long-range, low-power wireless network, made up of Sub-1 GHz CC13x0 or CC13x2 devices (both are supported) that run the TI 15.4-Stack-based application, can be connected to multiple *cloud service* providers, such as IBM Watson IoT®, AWS IoT, and so on. An online dashboard is provided, which allows users visualize the real-time sensor data as well as send actuation commands from anywhere in the world using an Internet-connected device with a web browser.

This reference design provides a list of suggested hardware, schematics, and foundational software to quickly begin IoT product development. The design also provides the ability to visualize the data inside a local network without connecting to a *cloud service*. The software design is created to be flexible, to enable other *cloud service* providers of choice.

This reference design enables IoT in numerous applications, such as building security gateways, door and window sensor networks, asset management and tracking, and other IoT-enabled home and industrial automation applications.

The connection between the wireless sensor network and the cloud is made possible by the TI SimpleLink CC3220 device on the CC3220SF LaunchPad™ development platform. On one side, the CC3220 is connected to a Sub-1 GHz device acting as the central node in the wireless network, and on the other side, the device is connected to a cloud service such as IBM Watson IoT or AWS IoT using Wi-Fi®. These two connections allow the CC3220 device to act as a gateway to get the sensor messages from the Sub-1 GHz wireless network to the cloud and to get the actuation requests from the cloud dashboard sent back to the Sub-1 GHz wireless network.

Due to the long-range and low-power capabilities of the Sub-1 GHz sensors, this reference design may be useful for any application that would benefit from distributed sensing. This reference design provides an example that gives the ability to visualize or actuate tens or hundreds of sensors while only needing one gateway device, the SimpleLink CC3220, to be connected to the Internet.

2 System Overview

2.1 Block Diagram

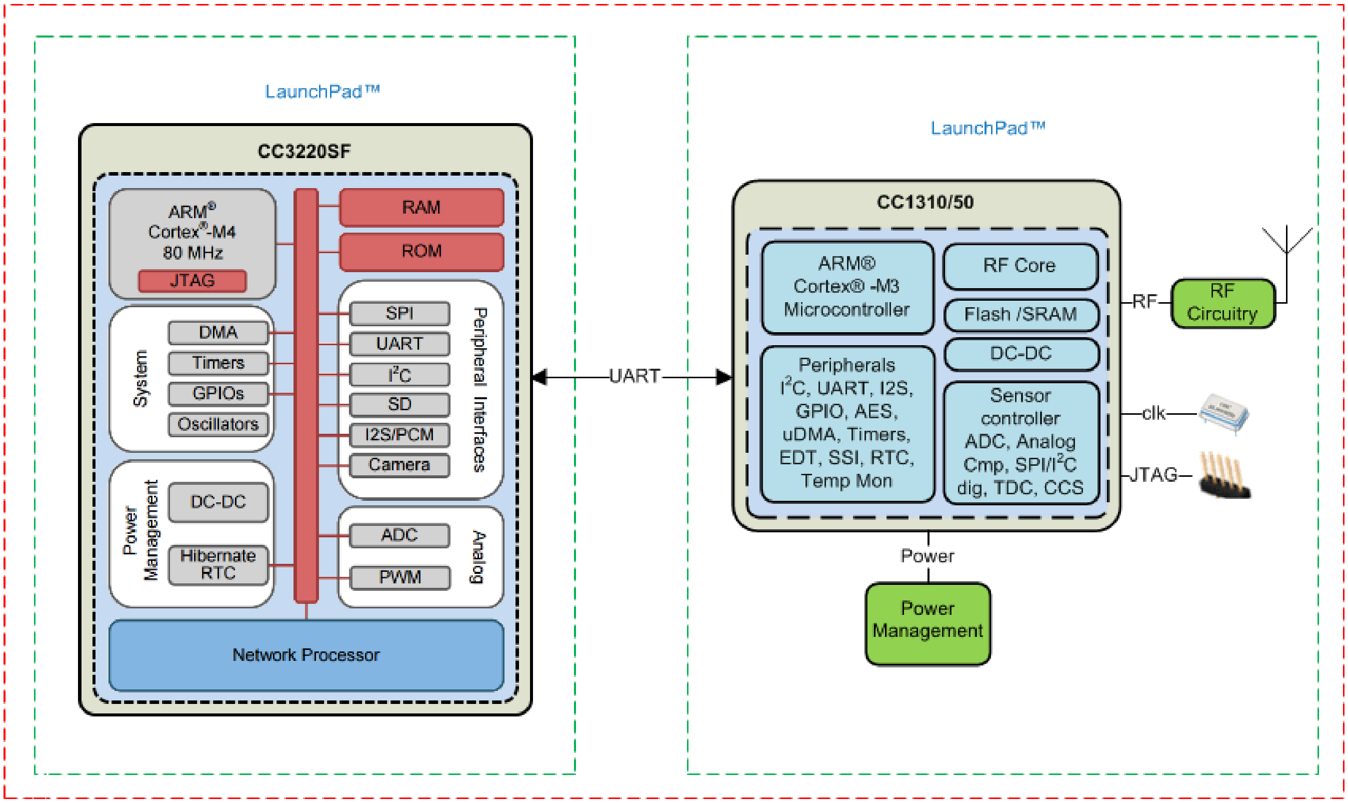
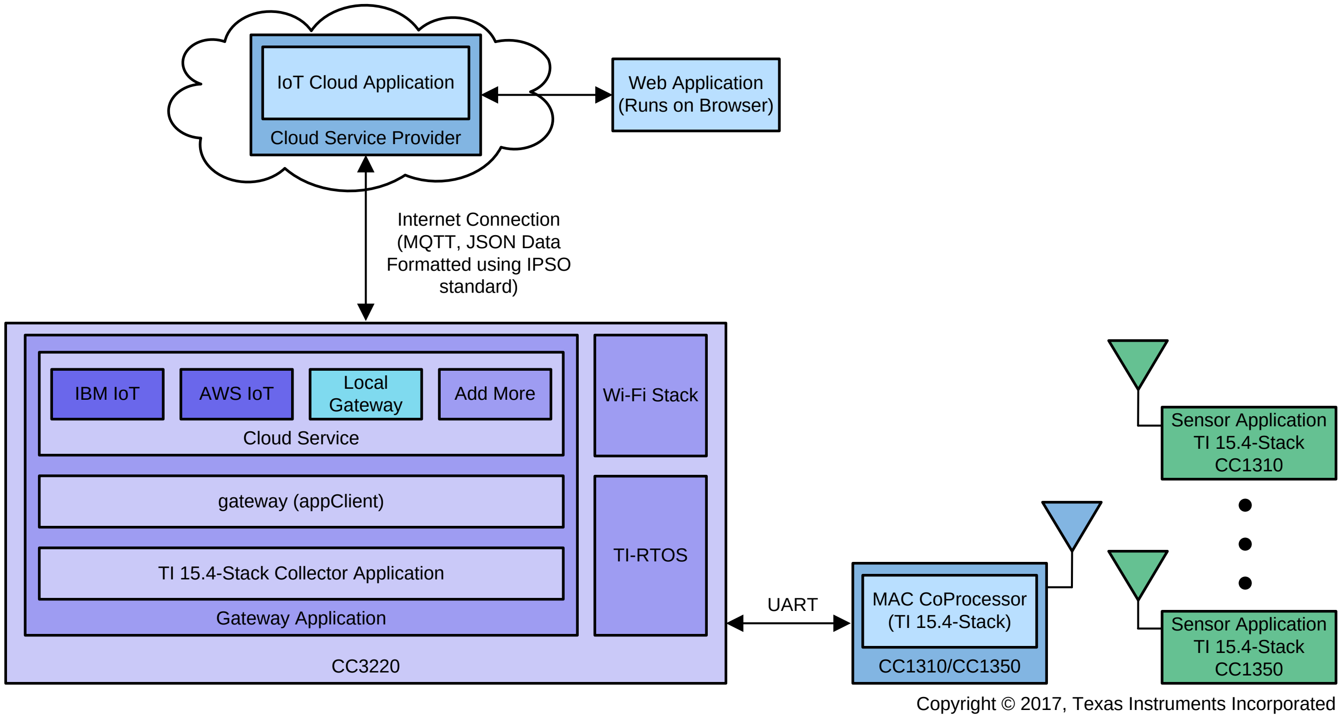


Figure 1. Block Diagram of IoT Gateway Reference Design

2.1.1 Software Block Diagram


IoT Cloud Application
Cloud Service Provider
Web Application (Runs on Browser)
Internet Connection (MQTT, JSON Data Formatted using IPSO standard)
IBM IoT
AWS IoT
Local Gateway
Add More
Cloud Service
Wi-Fi Stack
gateway (appClient)
TI-RTOS
TI 15.4-Stack Collector Application
Gateway Application
CC3220
UART
MAC CoProcessor (TI 15.4-Stack)
CC1310/CC1350
Sensor Application TI 15.4-Stack CC1310
Sensor Application TI 15.4-Stack CC1350

Copyright © 2017, Texas Instruments Incorporated

Figure 2. Software Block Diagram of TI 15.4-Stack Sensor-to-Cloud Reference Design

The following is a high-level description of each module in the software block diagram:

- User interface application: This application presents the network information and device information and provides ability to control network behavior to the end user.
- IoT cloud application: This application runs on the cloud server, which communicates with the IoT gateway application. The interface of the *cloud service* task with the cloud server is described in Section 2.3.1.
- IoT gateway application: This application runs on the SimpleLink CC3220. The application interfaces on one side with the *cloud service* task to enable cloud connectivity and on the other side to the *collector* task to interface with the TI 15.4-Stack based network. The interface between the IoT gateway and the *cloud service* is described in Section 2.3.1.
  - *Cloud service* task: This task provides the *cloud service* provider specific functionality. Users can take the current interface, which is designed as an extensible framework, and quickly modify the interface to add their own functionality for their end product development.
  - *Gateway*: This application component interfaces with the *collector* task through a POSIX message queue interface to enable connection with the TI-15.4 Stack network.
- *TI 15.4-Stack collector* task: This task implements the functionality that starts the network, allows new devices to join the network, configures the joining devices on how often to report the sensor data, configures how often to poll for buffered messages in case of non-beacon and frequency-hopping mode of network operation for sleepy network devices, and tracks connected devices to determine if they are active or inactive on the network. This determination is achieved by the collector periodically sending tracking request messages and awaiting corresponding tracking response messages. The *collector* task also implements components that talk to the *gateway* module. The communication is implemented through POSIX-based message queues.
- *MAC CoP* application: The MAC coprocessor application runs on the CC13x0 or CC13x2 LaunchPad, which provides a UART-based interface from TI 15.4-Stack to the IoT gateway application.
- CC13x0 or CC13x2 LaunchPad Sensor End Node: The sensor example application from TI 15.4-Stack and runs on the CC13x0 or CC13x2 LaunchPad.

2.2 Highlighted Products

This section highlights key hardware devices and software components used in the reference design.

2.2.1 SimpleLink™ CC13x0 and CC13x2

The CC13xx is a member of the SimpleLink family of cost-effective, ULP, 2.4-GHz and Sub-1 GHz RF devices. In addition to flexible low-power modes, very-low active RF and MCU current consumption provide excellent battery lifetime and allow long-range operation on small, coin-cell batteries and in energy-harvesting applications.

The CC13x2R devices combine a flexible, very low-power RF transceiver with a powerful 48-MHz Arm® Cortex®-M4F CPU in a platform supporting multiple physical layers and RF standards. A dedicated Radio Controller (Arm® Cortex®-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus ensuring ultra-low power and great flexibility. The low power consumption of the CC1352R device does not come at the expense of RF performance; the CC1352R device has excellent sensitivity and robustness (selectivity and blocking) performance.

The CC1350 is the first device in the CC13xx and CC26xx family of cost-effective, ULP wireless MCUs capable of handling both Sub-1 GHz and 2.4-GHz RF frequencies. The CC1350 device combines a flexible, very-low-power RF transceiver with a powerful, 48-MHz, Cortex®-M3 MCU in a platform supporting multiple physical layers and RF standards. A dedicated radio controller (Cortex-M0) handles low-level RF protocol commands that are stored in ROM or RAM, thus, ensuring ULP and flexibility to handle both Sub-1 GHz protocols and 2.4-GHz protocols (for example, *Bluetooth*® low energy). This enables the combination of a Sub-1 GHz communication stack that offers the best possible RF range together with a connection to a Bluetooth low energy smartphone that enables a great user experience through a phone application. The Sub-1 GHz-only devices in this family are the CC1310 and the CC1312.

2.2.2 SimpleLink™ CC3220

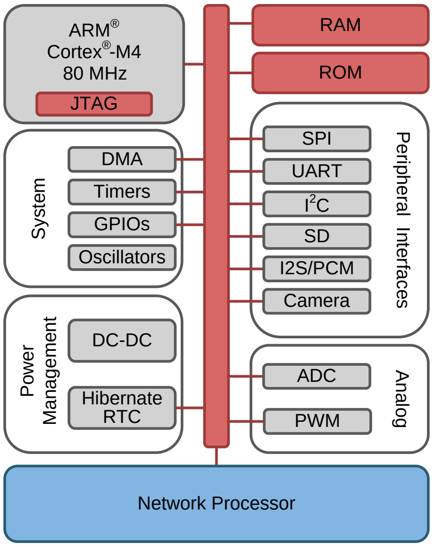
The CC3220x device is part of the SimpleLink MCU platform, which consists of Wi-Fi, low energy, Sub-1 GHz and host MCUs, which all share a common, easy-to-use development environment with a single core SDK and rich tool set. A one-time integration of the SimpleLink platform enables the user to add any combination of the portfolio's devices into their design, which allows 100% code reuse when the design requirements change. For more information, visit *SimpleLink Solutions* overview.

Created for the IoT, the SimpleLink CC3220x device family from Texas Instruments is a single-chip solution that integrates two physically separated, on-chip MCUs. One of the MCUs is an application processor— an ARM® Cortex®-M4 with a user-dedicated 256KB of RAM and an optional 1MB of XIP flash. The other MCU is a network processor in charge of running all Wi-Fi and Internet logical layers. This ROM-based subsystem includes an 802.11b/g/n radio, baseband, and MAC with a powerful crypto engine for fast, secure Internet connections with 256-bit encryption.

The CC3220x wireless MCU family is part of the second generation of TI's Internet-on-a-chip™ family. This generation introduces new features and capabilities that further simplify the connectivity of things to the Internet. The new capabilities including the following:

- IPv6
- Enhanced Wi-Fi provisioning
- Enhanced power consumption
- Enhanced file system security (supported only by the CC3220S and CC3220SF devices)
- Wi-Fi AP connection with up to four stations
- More concurrently opened BSD sockets; up to 16 BSD sockets, of which six are secure
- HTTPS support
- RESTful API support
- Asymmetric keys crypto library

The CC3220x wireless MCU family supports the following modes: station, AP, and Wi-Fi Direct®. The device also supports both WPA2-Personal and WPA2-Enterprise security modes. This subsystem includes embedded TCP/IP and TLS/SSL stacks, HTTP server, and multiple Internet protocols. The device supports a variety of Wi-Fi provisioning methods including HTTP based on AP mode, SmartConfig™ technology, and WPS2.0.



Copyright © 2017, Texas Instruments Incorporated

Figure 3. CC3220 Block Diagram

2.2.3 TI 15.4-Stack

TI 15.4-Stack is an IEEE802.15.4e/g-based software stack part of the SimpleLink CC13x0 SDK supporting a star network topology for Sub-1 GHz applications. TI 15.4-Stack software runs on the SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU from TI. The TI 15-4 Stack offers several key benefits, such as longer range in FCC band and better protection against in-band interference by implementing frequency hopping. The SDK also offers customers an accelerated time to market by providing a complete end-to-end, node-to-gateway reference design. TI 15.4-Stack is supported on the industry's lowest-power SimpleLink Sub-1 GHz wireless MCU platform.

This release is available royalty-free to customers using TI's CC13x0 or CC13x2 wireless MCU and also runs on TI's SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU LaunchPad development kit. This release is available royalty-free to customers using TI's CC13x0 or CC13x2 wireless MCU and also runs on the SimpleLink Sub-1 GHz CC13x0 or CC13x2 wireless MCU LaunchPad development kit from TI.

Features:

- IEEE 802.15.4e/g standards-based software stack
- Frequency hopping
- Medium access with CSMA/CA
- Built-in acknowledgment and retry
- Network and device management (joining, commissioning, service discovery)
- Security feature through AES 128-bit encryption and integrity check
- Supported on SimpleLink Sub-1 GHz CC1310 wireless MCU
- Star topology: point-to-point, one-to-many, and data concentrator
- Synchronous (beacon) and asynchronous (non-beacon) modes
- Designed for 915-MHz FCC, 863-MHz ETSI, and 433-MHz China bands
- SimpleLink long range mode for all supported frequency bands
- Support for SimpleLink CC1190

- Bluetooth low energy beacon advertisement support
- Sensor-to-web example application
- Easy application development guided through sample applications showcasing the stack configuration and APIs
- Coprocessor mode for adding connectivity to any MCU or MPU with Linux® host middleware and console application

For more details and to get the TI 15.4-Stack software, download the SimpleLink CC13x0 SDK, which includes the TI 15.4-Stack.

2.2.4 SimpleLink™ Wi-Fi CC3220 SDK

The SimpleLink Wi-Fi CC3220 SDK contains drivers for the CC3220 programmable MCU, over 30 sample applications, and related documentation. The SDK also contains the flash programmer, a command line tool for flashing software, configuring network and software parameters (SSID, access point channel, network profile, and so on), system files, and user files (certificates, web pages, and so on). This SDK can be used with TI's SimpleLink Wi-Fi CC3220 LaunchPad development kits.

Features:

- Internet-on-a-chip sample applications:
  - Email from SimpleLink Wi-Fi
  - Information center: Get time and weather from the Internet
  - https server: Host a secure web page on SimpleLink Wi-Fi
  - XMPP: IM chat client
  - Serial interface
- Wi-Fi sample applications:
  - Easy Wi-Fi configuration
  - Station, AP modes
  - TCP/UDP
  - Security—Enterprise and personal, TLS/SSL
  - Power management—Deep sleep, hibernate
- MCU peripheral sample applications:
  - Including parallel camera, I2S audio, ADC, I^{2}C, PWMs, JTAG Flashing, and more

2.3 System Design Theory

2.3.1 TI IoT Gateway-to-Cloud Service Interface

The purpose of this section is to provide a description of the message types and expected data flows that will be shared between the TI IoT gateway and an IoT cloud server. The interface is designed to be flexible to support multiple cloud vendors. For this purpose, the Sub-1 GHz wireless network and node information will be exchanged between the gateway and the cloud using the long-established JavaScript object notation (JSON) format. Additionally, IPSO alliance smart object definitions will be used to define sensors (and their data) that are connected to each node in the wireless networks.

2.3.1.1 *Message Types*

To fully specify the Sub-1 GHz wireless network information, as well as the Sub-1 GHz sensors and their data, two distinct message types have been defined for the IoT gateway to update the cloud. In order to allow the cloud to send messages back to the TI IoT gateway, two additional message types are defined that allow the cloud to update the wireless network state and send actuation messages to specific devices in the network.

2.3.1.1.1 Network Information Message Type (From TI IoT Gateway to the Cloud)

This message type presents information about the wireless network, its current state, and a list of devices that are connected to the network. As described later in this design guide, this will be the first message type sent after the network is initialized.This message type contains all the information necessary to prepare for receiving sensor data from devices. This message type contains the following fields:

- name: begins as the short address of the network but allows for the cloud to provide a more specific name
- channels: list of channels that the wireless network is operating on
- pan_id: the 16-bit PAN identifier of the network
- short_addr: the 16-bit short address of the PAN-coordinator
- ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
- security_enabled: yes, if security enabled; no, otherwise
- mode: network operation mode (beacon, non-beacon, frequency hopping)
- state: PAN-coordinator state values (waiting, starting, restoring, started, open, closed)
- devices: list of wireless nodes in the network
  - name: begins as the short address of the device but allows cloud to update
  - short_addr: the 16-bit short address of the PAN-coordinator
  - ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
  - topic: the topic that the device will send its sensor data updates to
  - object_list: list of IPSO alliance smart objects (sensors) attached to this device
    - oid: object ID which specifies the sensor type in the IPSO standard
    - iid: list of instance IDs for the current object (can be multiple same type sensors)

2.3.1.1.2 Device Information Message Type (From TI IoT Gateway to the Cloud)

This message type provides information about the wireless device as well as the latest data for all of the sensors connected to the device. This message type will be sent when a device reports sensor data or switches between an active or inactive state. The following fields are contained in this message type:

- active: whether or not the wireless node is active
- ext_addr: the 64-bit IEEE extended address of the PAN-coordinator device
- rssi: received signal strength indicator of the last message received
- smart_objects: list of the IPSO alliance smart objects connected to this wireless device
  - object ID description: type of sensor (as defined in the IPSO standard); can be multiple types of sensors connected to each device
  - instance ID: the instance ID for the parent object type; can be multiple sensors of the same type
  - resource ID description list: sensor data name value pairs (for example, *sensorValue*: 32.5, *units:* Celsius, and so forth); these resources match what is specified for the given object ID in the IPSO standard

2.3.1.1.3 Update Network State Message Type (From Cloud to TI IoT Gateway)

In the current implementation of the TI IoT gateway, this message type is intended to be able to open or close the wireless network to new devices joining. The cloud's front end user interface can allow a user to click a button to open or close the network and then generate this message type and send it to the TI IoT gateway. The gateway will then notify the network on whether it needs to open or close to new device joins. This message type only includes the desired state of the network and should be sent to the same topic that the cloud is receiving the network information messages from. The following field is all that is required:

- state: should be set to either *open* or *closed*

2.3.1.1.4 Device Actuation Message Type (From Cloud to TI IoT Gateway)

This message type is added to allow the cloud to send actuation messages to specific devices in the wireless network. The current implementation only supports toggling an LED on the wireless device's board. The device actuation message should be sent to the topic of the device as given in the devices list of the network information message. The following field is the only requirement for this message:

- toggleLED: should be set to *true*

2.3.1.2 Data Flows

2.3.1.2.1 Network Information Sent to the Cloud

The following bulleted items are the list of events that can occur on the TI IoT gateway that will cause a network information message type to be sent to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

- **Network Startup**

  This is the initial event in the TI IoT gateway. The TI IoT gateway will aggregate the information about the wireless network as well as the list of connected devices and their sensor types. The TI IoT gateway will then make a connection to the cloud and send the aggregated data contained in the network information message type.
- **Network Information Update**

  This event can occur if any of the information about the wireless network changes. For example, if the network operation mode of the wireless network was changed, the TI IoT gateway would once again aggregate all the information needed (network information and device list) and send the network information message type to the cloud.
- **Network State Change**

  This event occurs if the state of the wireless network changes. For example, if the network state changes from open to closed the TI IoT gateway will send a network information message type to the cloud.
- **Device Joins the Wireless Network**

  When a new device joins the network, after the network is up and running, this event will occur. In this case, the TI IoT gateway will add the new device and its information to the devices list within the network information message type and then send the updated information to the cloud.
- **Expected Cloud Behavior**

  It is expected that the cloud will be prepared for the network startup event and will be able to receive the network information message type (using a wildcard and then filtering or by having prior knowledge about the destination or topic of the message). Once the cloud receives the network information message, the wireless network information (PANID, security, mode, and so on) can be displayed to users and the device list information (topic, object list, and so on) can be used to prepare itself to receive and display device and sensor data.

2.3.1.2.2 Device Information Sent to the Cloud

The following bulleted items are the list of events that will cause the TI IoT gateway to send a device information message type to the cloud. A description is given with each event and the end of this section describes the expected behavior from the cloud upon receipt of this type of message.

- **Device Becomes Inactive**

  This event occurs when the TI IoT gateway detects that one of the devices in the connected devices list has stopped sending sensor data updates. The TI IoT gateway will update the *active* field and send a Device Information Message Type to the cloud for the inactive device.
- **Device Reports Sensor Data**

  Each time a sensor on a connected device reports sensor data this event occurs. The TI IoT gateway updates the IPSO Alliance Smart Object list in the device for each sensor and then sends a device information message type to the cloud.
- **Expected Cloud Behavior**

It is expected that the cloud will be listening on each topic given in the connected devices list from the network information message. When one of the two events occur in this section, the TI IoT gateway will send the device information message to the topic (corresponding to the device being update) that the cloud should be listening on or subscribed to. When the device information message arrives at the cloud, the cloud should display the latest device information and sensor data to users.

2.3.1.2.3 Update Network State Message Sent to the TI IoT Gateway

This message is used to open or close the wireless network to new devices joining. This message should be an option provided to users in the front end user interface that the cloud presents. When the user decides to update the network state, the cloud should send an update network state message type to the TI IoT gateway on the same topic that the network information messages are arriving on.

- **Expected TI IoT Gateway Behavior**

  The TI IoT gateway will receive the Update Network State message and will generate the correct command (either open or close) to the wireless network. This message should in turn cause a network state change event (from 7.2.1 above) that will send a network information message back to the cloud which can confirm the successful completion of the Update Network State command.

2.3.1.2.4 Device Actuation Message Sent to the TI IoT Gateway

This method is used to toggle the LED on the board of the connected devices. This message is meant to be a proof-of-concept on the current device setup and will change for customer use-case specific actuations. A toggle LED button for each device will be provided to users of the cloud's front end interface. When the toggle LED button is clicked the cloud should send a device actuation message to the TI IoT gateway on the same topic that the device information messages are arriving on.

- **Expected TI IoT Gateway Behavior**

  The TI IoT gateway will generate a toggle LED command and send it to the device corresponding to the topic that the device actuation message was received on. This will cause the LED to toggle. Because the state of the LED is not captured in the device information message type, there will be no feedback to the cloud that the LED actually toggled.

3 Hardware, Software, Testing Requirements, and Test Results

3.1 Required Hardware and Software

The CC3220SF plus CC13x0/CC13x2 sensor-to-cloud reference design helps developers create ULP, long-range, star-topology network solutions. The sensor-to-cloud reference design includes the Gateway example application running on the CC3220SF, MAC CoProcessor (CoP) application running on the CC13x0 or CC13x2, in addition to sensor node applications. The CC3220SF Gateway example application interfaces over UART with a CC13x0 or CC13x2 LaunchPad, which acts as a MAC CoP. The Gateway example application implements a IEEE 802.15.4 full-function device, which performs the functions of a network PAN coordinator (starting a network and permitting devices to join this network) and also provides an interface for monitoring and collecting sensor data from one or more sensor devices.

The Gateway example application provides an IEEE 802.15.4 network to IP bridge and is a great starting point to create IoT applications based on TI 15.4-Stack.

3.1.1 Hardware

- 2× CC13x0 or 2× CC13x2 LaunchPad development kits
- 1× CC3220SF LaunchPad development kit
- USB cables
- Wi-Fi access point with Internet access

3.1.2 Software

- CC3220-SensorToCloud SW
- CC3220 SDK v1.50.00.06
- SimpleLink CC13x0 SDKor SimpleLink CC13x2 SDK
- UniFlash v4.1.1.1250 or later
- Tera Term or any other equivalent terminal program
- Cloud Foundry CLI (for IBM Cloud only)
- (Optional) SimpleLink Starter Pro IOS® app or SimpleLink Wi-Fi Starter Pro Android™ app (downloaded from the app store on smartphones or tablets)

3.2 Testing and Results

This section describes the hardware and software used for running the tests and the results obtained.

3.2.1 Test Setup

During the development process of this reference design, the full hardware and software portions described in earlier sections were used for testing. Multiple CC13x0 and CC13x2 sensor nodes and a CC3220SF LaunchPad (connected to a CC1310 coprocessor) were used to verify the IoT gateway functionality with the IBM cloud and AWS IoT cloud services. The test results of this reference design can be visualized by the IoT dashboard shown in Section 3.2.2.

3.2.1.1 Running the Out-of-Box Examples

This section provides detailed instructions to assist developers set up and understand the principles behind the out-of-box examples.

Some of the topics covered in this section are:

1. Programming the CC13xx LaunchPad development kits
2. Programming the gateway example application on the CC3220SF LaunchPad
3. Connecting the CC13x0 *MAC-CoP* LaunchPad with the CC3220SF LaunchPad
4. Setting up and configuring the cloud service
5. Running and using the example

For steps on how to setup and run the *IBM gateway example* please follow the instructions in Section 3.2.1.1.1 and to setup and run the *AWS IoT gateway example* please follow the instructions in Section 3.2.1.1.2.

NOTE: This guide can be performed using either CC13x0 or CC13x2 LaunchPad development kits for the MAC CoP. The training material is based on CC1350, and the same procedures apply when using either a CC1310 or CC13x2 devices.

3.2.1.1.1 Running IBM Gateway Example

1. Label one CC1350 LaunchPad as *Sensor* and the other as *MAC-CoP*. These labels will be referred to throughout this guide. It is recommended to use non-permanent marking for this (for example, sticky notes) as these labels may only be relevant for this specific example.
2. Program the *MAC-CoP* and the *Sensor* LaunchPad by following the steps in Section 3.2.1.1.3.
3. Program the CC3220SF LaunchPad by following the steps in Section 3.2.1.1.4.
4. Connect the MAC-CoP LaunchPad with the CC3220SF LaunchPad by following the steps in Section 3.2.1.1.5.
5. Set up an account for IBM Cloud by following the instructions in Section 3.2.1.1.6.
6. Set up the Watson IoT platform service as in Section 3.2.1.1.7.
7. Follow Section 3.2.1.1.8 to set up Node.js and cloud foundry app.
8. After all of these steps have been completed, go to Section 3.2.1.1.9, and follow the steps to connect the gateway to the Internet and run the example.

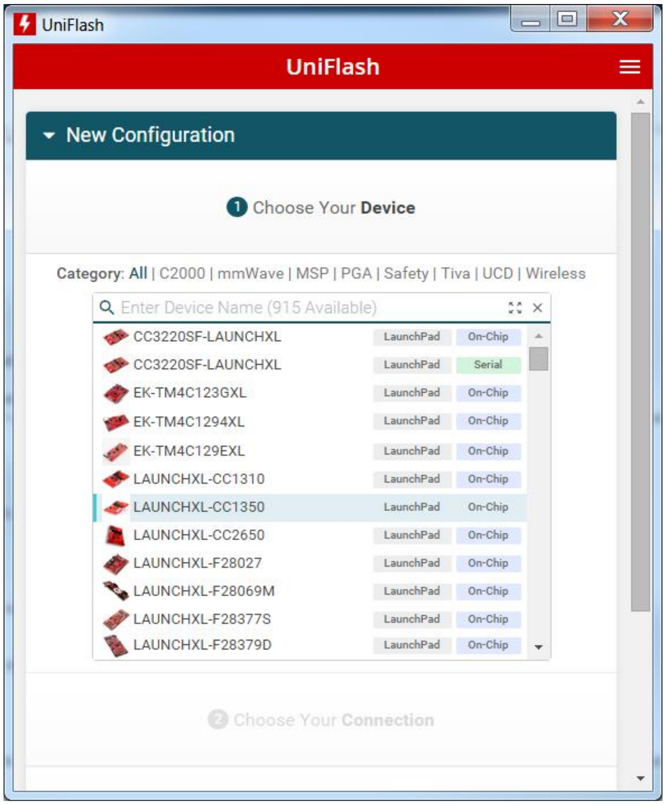
3.2.1.1.2 Running AWS IoT Example

Before getting started with the AWS IoT example please request the AWS IoT demo configuration from StackArmor by filling out the following form.

1. Label one CC1350 LaunchPad as Sensor and the other as MAC-CoP. These labels will be referred to throughout this guide. It is recommended to use non-permanent marking for this (for example, sticky notes), as these labels may only be relevant for this specific example.
2. Program the *MAC-CoP* and the Sensor *LaunchPad*™ by following the steps in Section 3.2.1.1.3.
3. Import and build the AWS IoT example by following the steps in Section 3.2.1.1.4.2.
4. Program the CC3220SF LaunchPad by following the steps in Section 3.2.1.1.4.4.
5. Connect the MAC-CoP LaunchPad with the CC3220SF LaunchPad by following the steps in Section 3.2.1.1.5.
6. After all of these steps have been completed, go to Section 3.2.1.1.9, and follow the steps on how to connect the gateway to the Internet and run the example.

3.2.1.1.3 Programming CC13x0 LaunchPad™

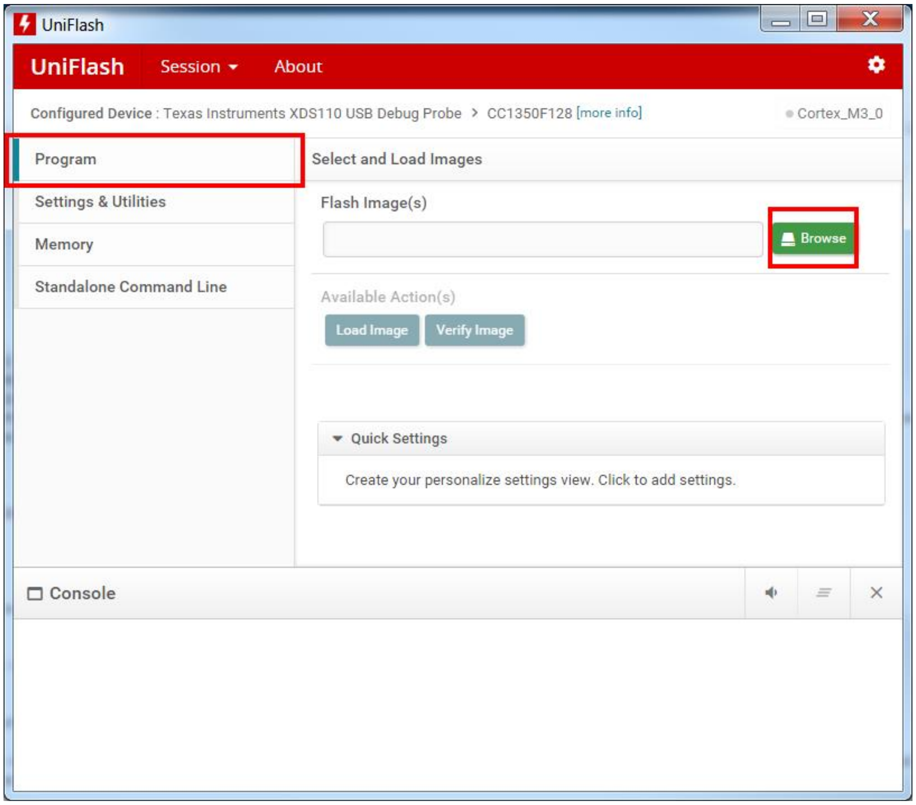
1. It is assumed that all the required software has already been installed. If not, install the required softwareSection 3.1.2 now.
2. Connect the *CC13x0* LaunchPad to the PC.
3. Open UniFlash.
4. Select LAUNCHXL-CC1350 as show in Figure 4, and click on the *Start* button.


UniFlash
UniFlash
New Configuration
1 Choose Your Device
Category: All | C2000 | mmWave | MSP | PGA | Safety | Tiva | UCD | Wireless
Enter Device Name (915 Available)
2 Choose Your Connection

CC3220SF-LAUNCHXL	LaunchPad	On-Chip
CC3220SF-LAUNCHXL	LaunchPad	Serial
EK-TM4C123GXL	LaunchPad	On-Chip
EK-TM4C1294XL	LaunchPad	On-Chip
EK-TM4C129EXL	LaunchPad	On-Chip
LAUNCHXL-CC1310	LaunchPad	On-Chip
LAUNCHXL-CC1350	LaunchPad	On-Chip
LAUNCHXL-CC2650	LaunchPad	On-Chip
LAUNCHXL-F28027	LaunchPad	On-Chip
LAUNCHXL-F28069M	LaunchPad	On-Chip
LAUNCHXL-F28377S	LaunchPad	On-Chip
LAUNCHXL-F28379D	LaunchPad	On-Chip

Figure 4. Choose Device: UniFlash

5. Make sure the *Program* tab is selected on the left, and click the *Browse* button to select the desired image for the CC13xx LaunchPad located in <S2C_Repo_Directory>\tidc01002\prebuilt\.



Configured Device : Texas Instruments XDS110 USB Debug Probe > CC1350F128 [more info]

Create your personalize settings view. Click to add settings.

Figure 5. Browse for Firmware Image

6. After selecting the desired image, click on the *Load Image* button to flash the CC1350 LaunchPad.

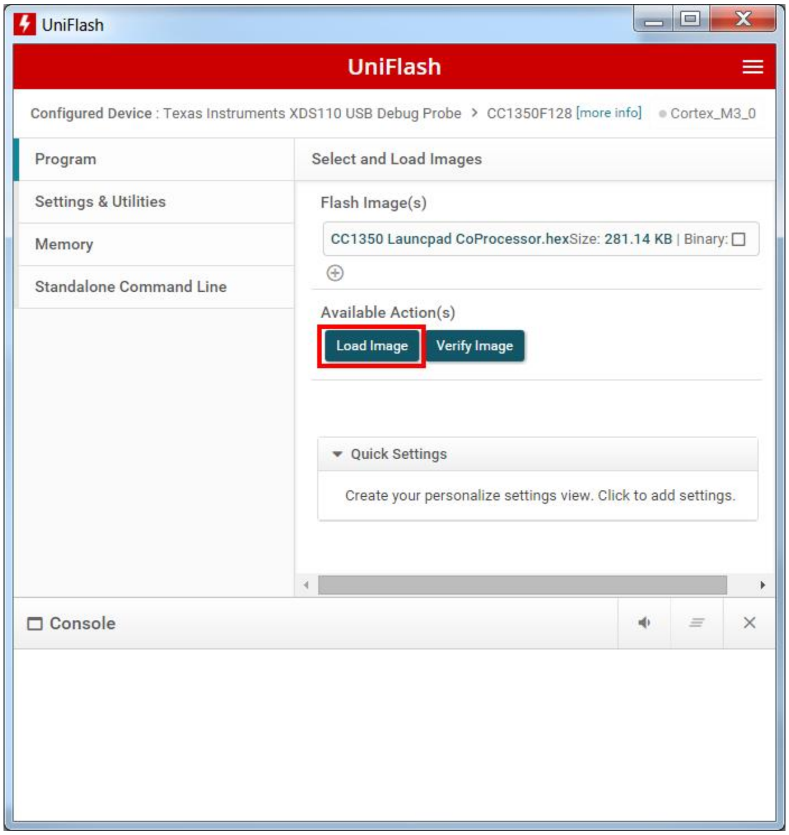
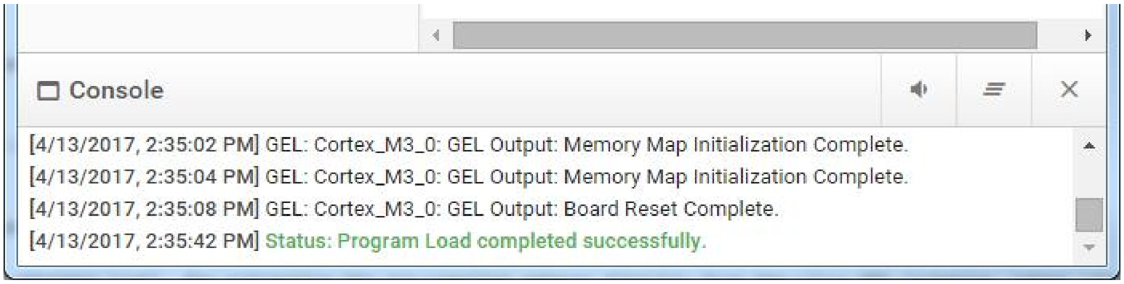

UniFlash
UniFlash
Configured Device : Texas Instruments XDS110 USB Debug Probe > CC1350F128 [more info] Cortex_M3_0
Program
Settings & Utilities
Memory
Standalone Command Line
Select and Load Images
Flash Image(s)
CC1350 Launcpad CoProcessor.hexSize: 281.14 KB | Binary: □
Available Action(s)
Load Image
Verify Image
Quick Settings
Create your personalize settings view. Click to add settings.
Console

Figure 6. Load Image

7. If loading the image was successful, a success message should show in the console as shown in Figure 7.



```
[4/13/2017, 2:35:02 PM] GEL: Cortex_M3_0: GEL Output: Memory Map Initialization Complete.
[4/13/2017, 2:35:04 PM] GEL: Cortex_M3_0: GEL Output: Memory Map Initialization Complete.
[4/13/2017, 2:35:08 PM] GEL: Cortex_M3_0: GEL Output: Board Reset Complete.
[4/13/2017, 2:35:42 PM] Status: Program Load completed successfully.
```

Figure 7. Successful Load

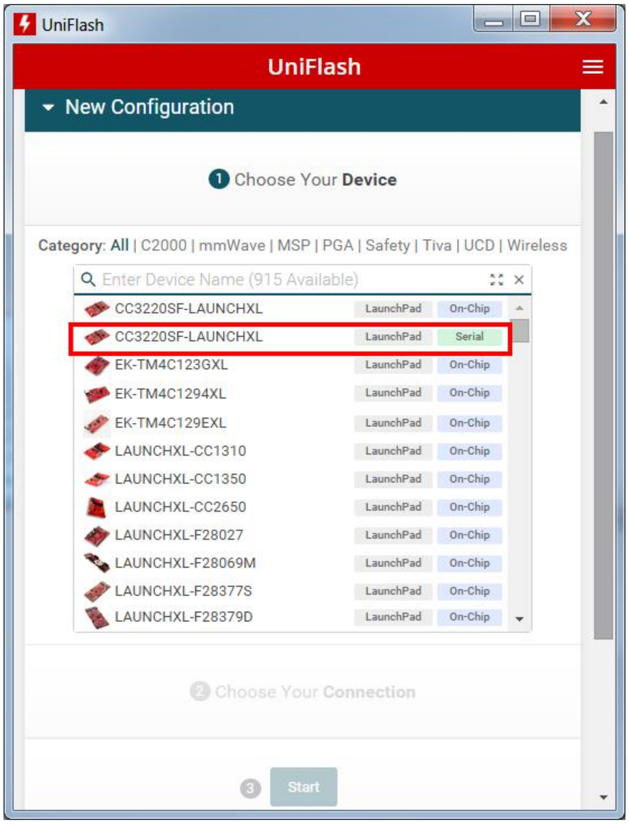
3.2.1.1.4 Programming the CC3220SF LaunchPad™

This section describes two ways of programming the CC3220SF LaunchPad. Section 3.2.1.1.4.1 explains the process of programming the CC3220SF LaunchPad by importing a preconfigured Image Creator project to UniFlash. Section 3.2.1.1.4.4 shows how to create an Image Creator project from scratch to program the CC3220SF with a binary generated from Code Composer Studio™ (CCS).

3.2.1.1.4.1 Programming a Preconfigured Image Creator Project

Note that only the IBM cloud service is provided in the preconfigured Image Creator project since the AWS IoT example needs extra configuration that need to be added and compiled in the source code.

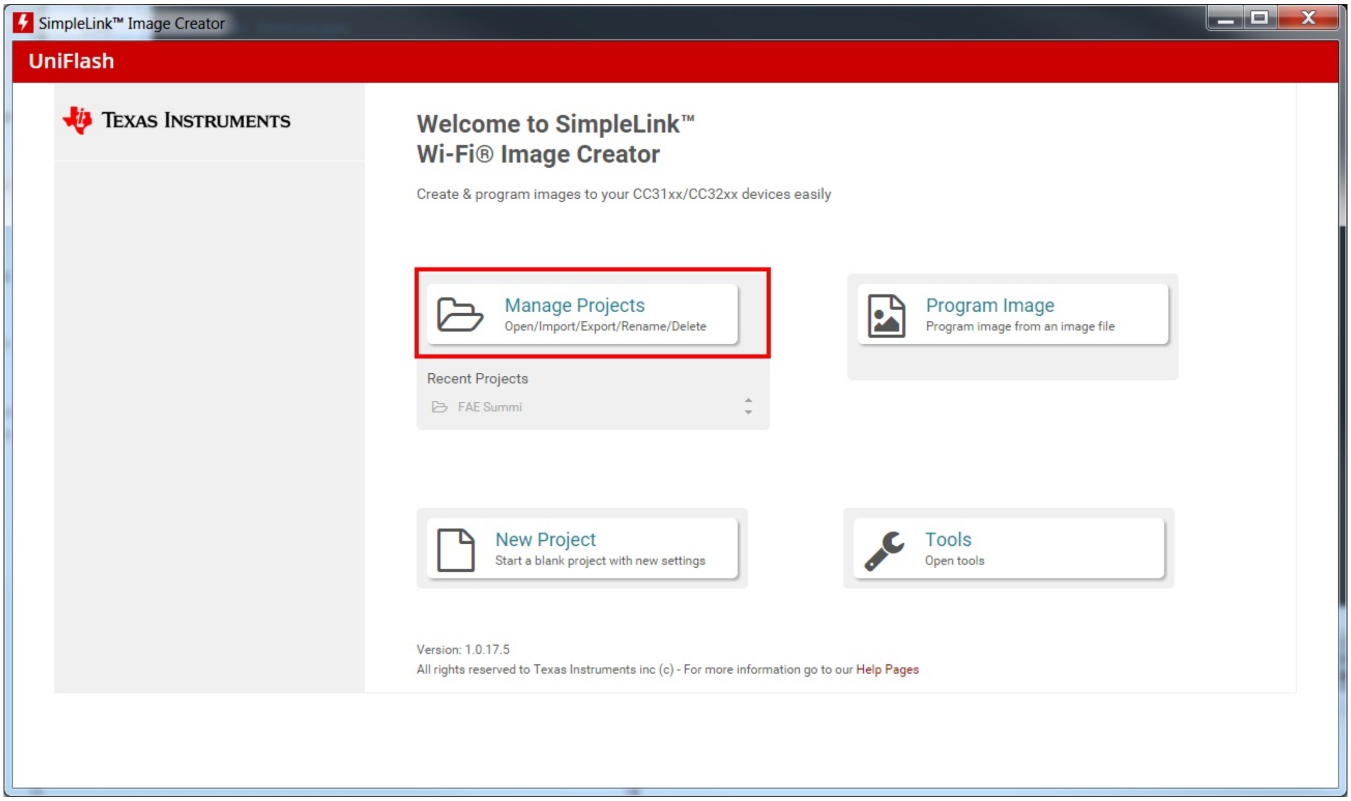
1. Open UniFlash.
2. On the *Choose your Device* section select *CC3220SF-LAUNCHXL*, and make sure to select the *Serial* option and not *On-Chip* as shown in Figure 8. Click on the *Start Image Creator* button.


UniFlash
UniFlash
New Configuration
1 Choose Your Device
Category: All | C2000 | mmWave | MSP | PGA | Safety | Tiva | UCD | Wireless
Enter Device Name (915 Available)
2 Choose Your Connection
3 Start

CC3220SF-LAUNCHXL	LaunchPad	On-Chip
CC3220SF-LAUNCHXL	LaunchPad	Serial
EK-TM4C123GXL	LaunchPad	On-Chip
EK-TM4C1294XL	LaunchPad	On-Chip
EK-TM4C129EXL	LaunchPad	On-Chip
LAUNCHXL-CC1310	LaunchPad	On-Chip
LAUNCHXL-CC1350	LaunchPad	On-Chip
LAUNCHXL-CC2650	LaunchPad	On-Chip
LAUNCHXL-F28027	LaunchPad	On-Chip
LAUNCHXL-F28069M	LaunchPad	On-Chip
LAUNCHXL-F28377S	LaunchPad	On-Chip
LAUNCHXL-F28379D	LaunchPad	On-Chip

Figure 8. Choose Device: CC3220SF

3. After starting Image Creator, click on the *Manage Projects* button as shown in Figure 9.


SimpleLink™ Image Creator
UniFlash
Texas Instruments
Manage Projects
Open/Import/Export/Rename/Delete
Recent Projects
FAE Summi
Program Image
Program image from an image file
New Project
Start a blank project with new settings
Tools
Open tools

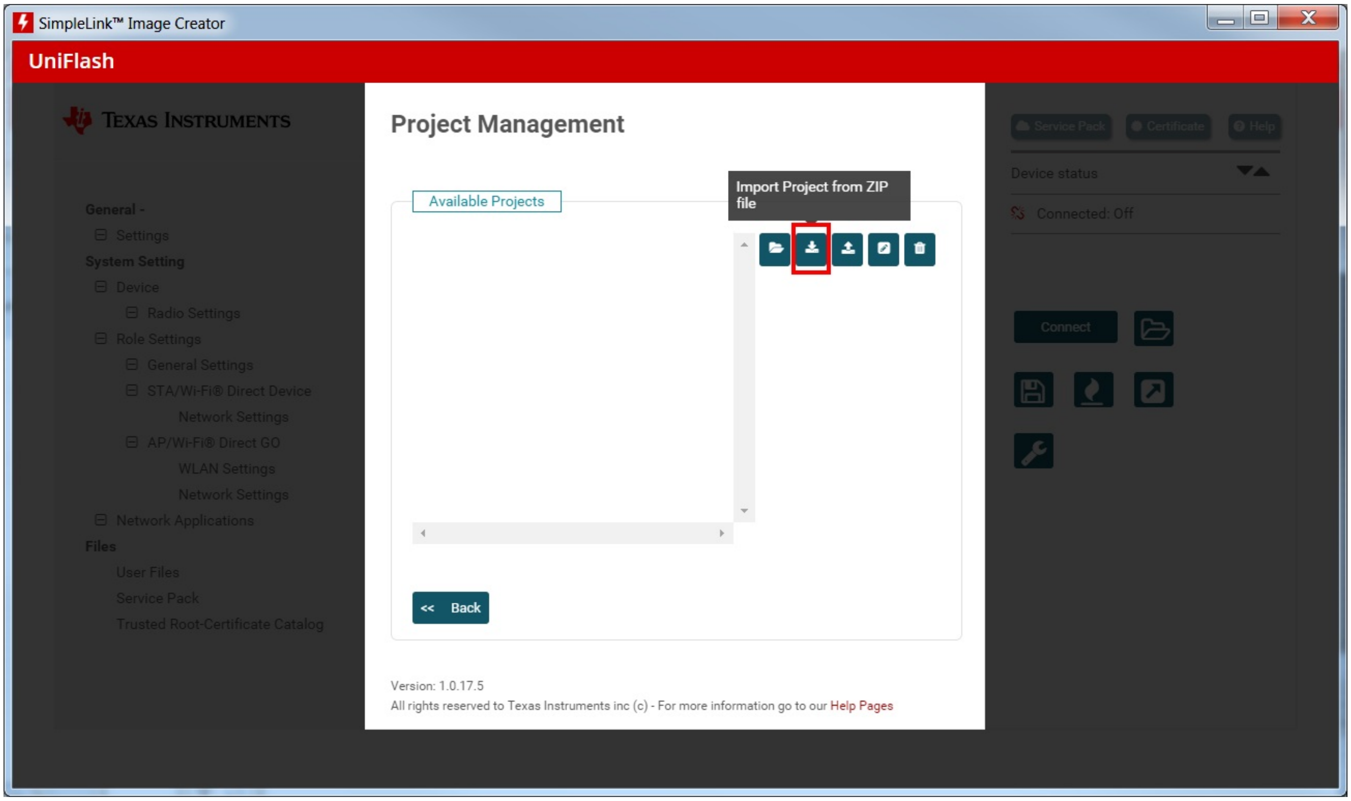
Welcome to SimpleLink™
Wi-Fi® Image Creator

Create & program images to your CC31xx/CC32xx devices easily

Version: 1.0.17.5
All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 9. Manage Project

4. Click on the *Import Project from ZIP file* button, and select the zip folder *C:\<S2C_Repo_Directory>\tidc01002\prebuilt\CC3220SF_LaunchXL\Uniflash_CC3220ImageCreatorProject.zip*.



Version: 1.0.17.5
All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 10. Import Project

5. Open the *CC3220SF_154StackGtway* project.

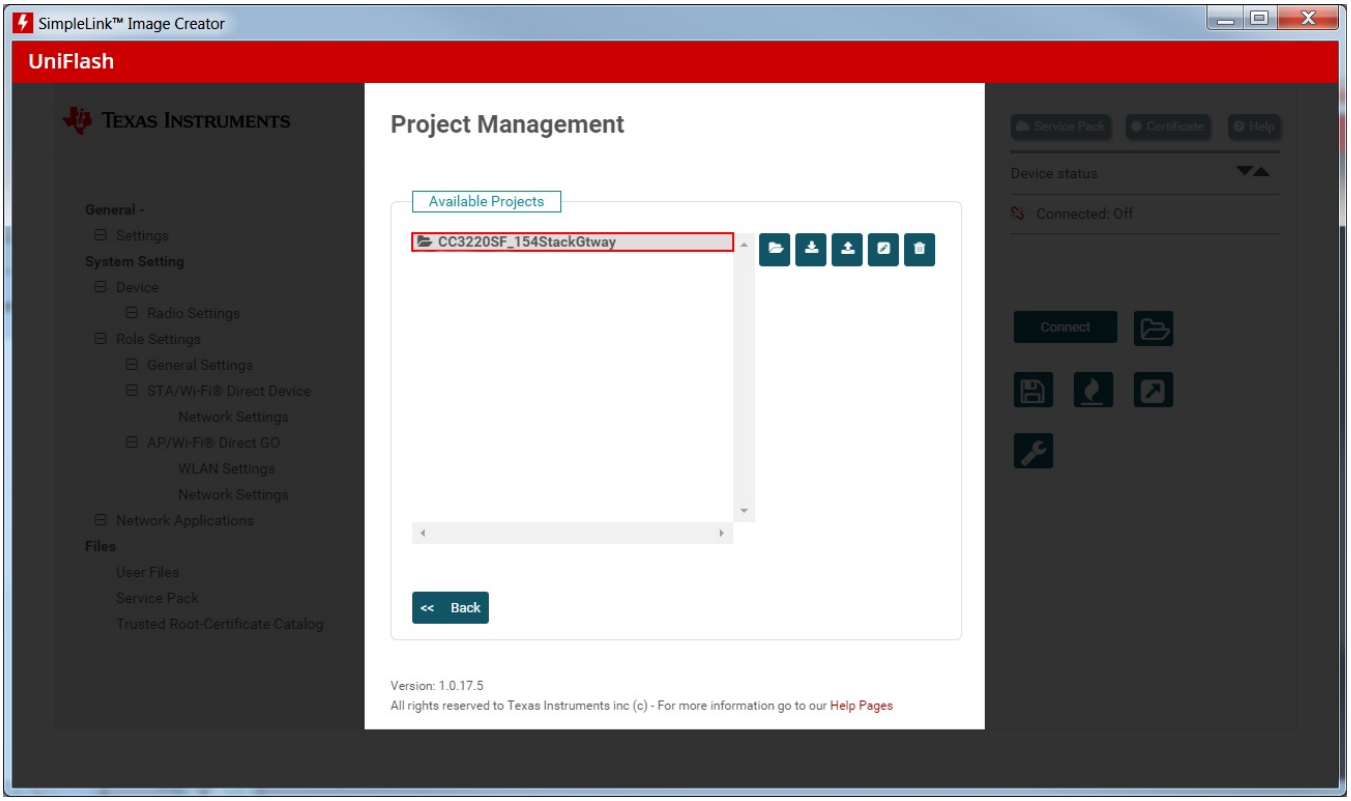
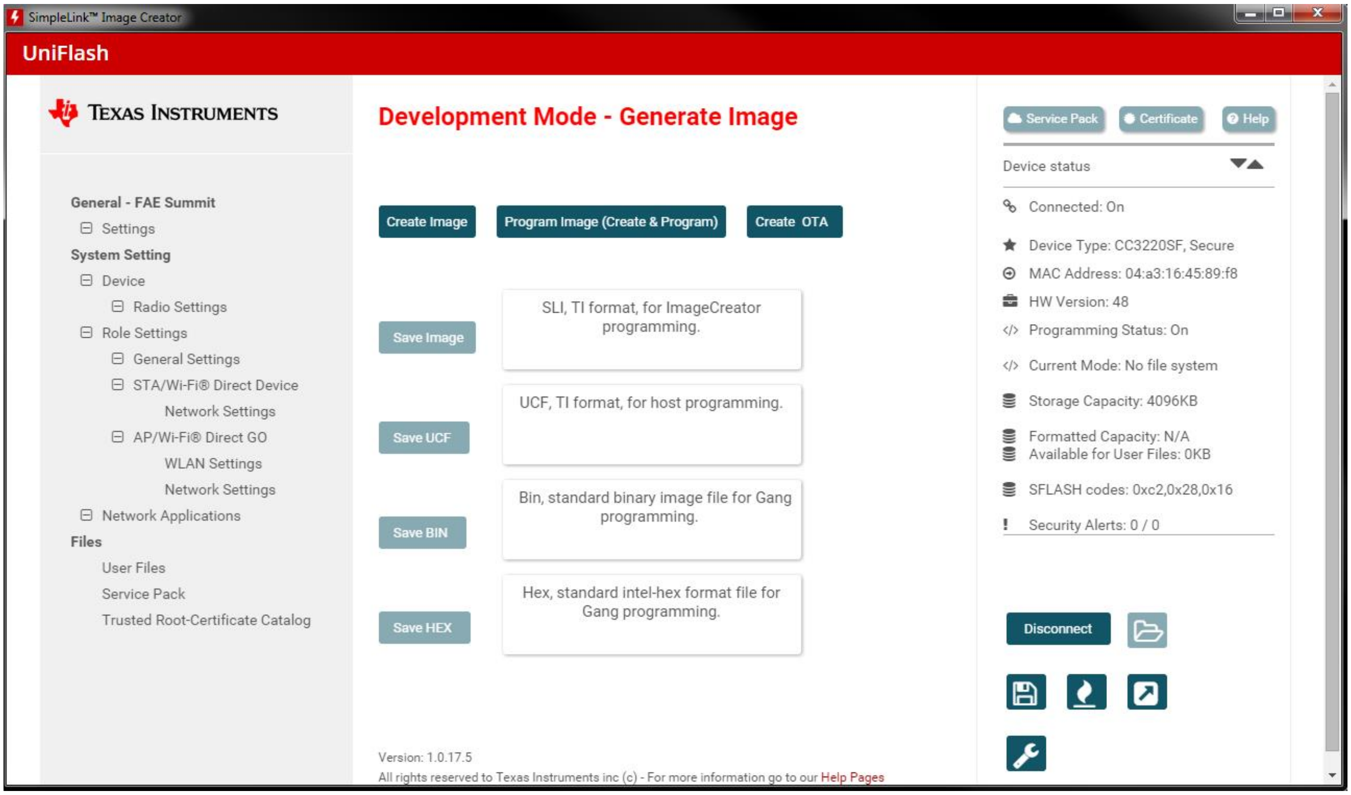


Figure 11. Open Project

6. Connect the device to the PC through a USB cable, and press the *Connect* button found on the bottom-right corner. Once the device is connected, select the *Generate Image* button underneath the *Disconnect* button, and select *Program Image (Create & Program)*.



SLI, TI format, for ImageCreator programming.

UCF, TI format, for host programming.

Bin, standard binary image file for Gang programming.

Hex, standard intel-hex format file for Gang programming.

Device status

Connected: On

Device Type: CC3220SF, Secure

MAC Address: 04:a3:16:45:89:f8

HW Version: 48

Programming Status: On

Current Mode: No file system

Storage Capacity: 4096KB

Formatted Capacity: N/A
Available for User Files: 0KB

SFLASH codes: 0xc2,0x28,0x16

Security Alerts: 0 / 0

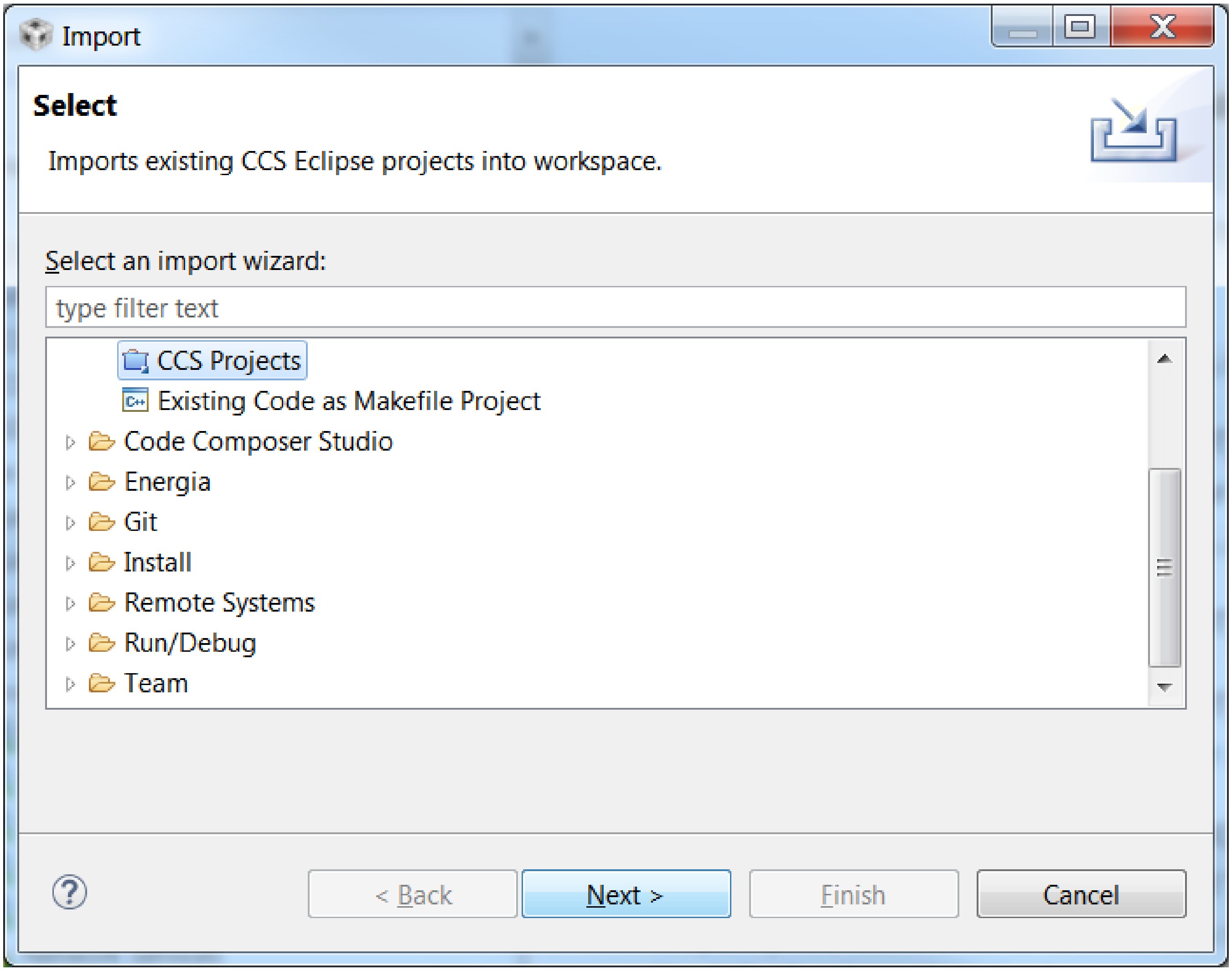
Version: 1.0.17.5

All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 12. Generate Image

3.2.1.1.4.2 Importing Examples to Code Composer Studio (CCS)

1. Open CCS version 7.3.
2. It is assumed that the CC3220 SDK has already been installed in a directory referred to as *CC32XX_SDK_INSTALL_DIR*. If not, install the CC3220 SDK now as this example requires it.
3. Import the Gateway project by going to *File* → *Import* → *C/C++* → *CCS Projects*, and click on *Next*.



Import

Select

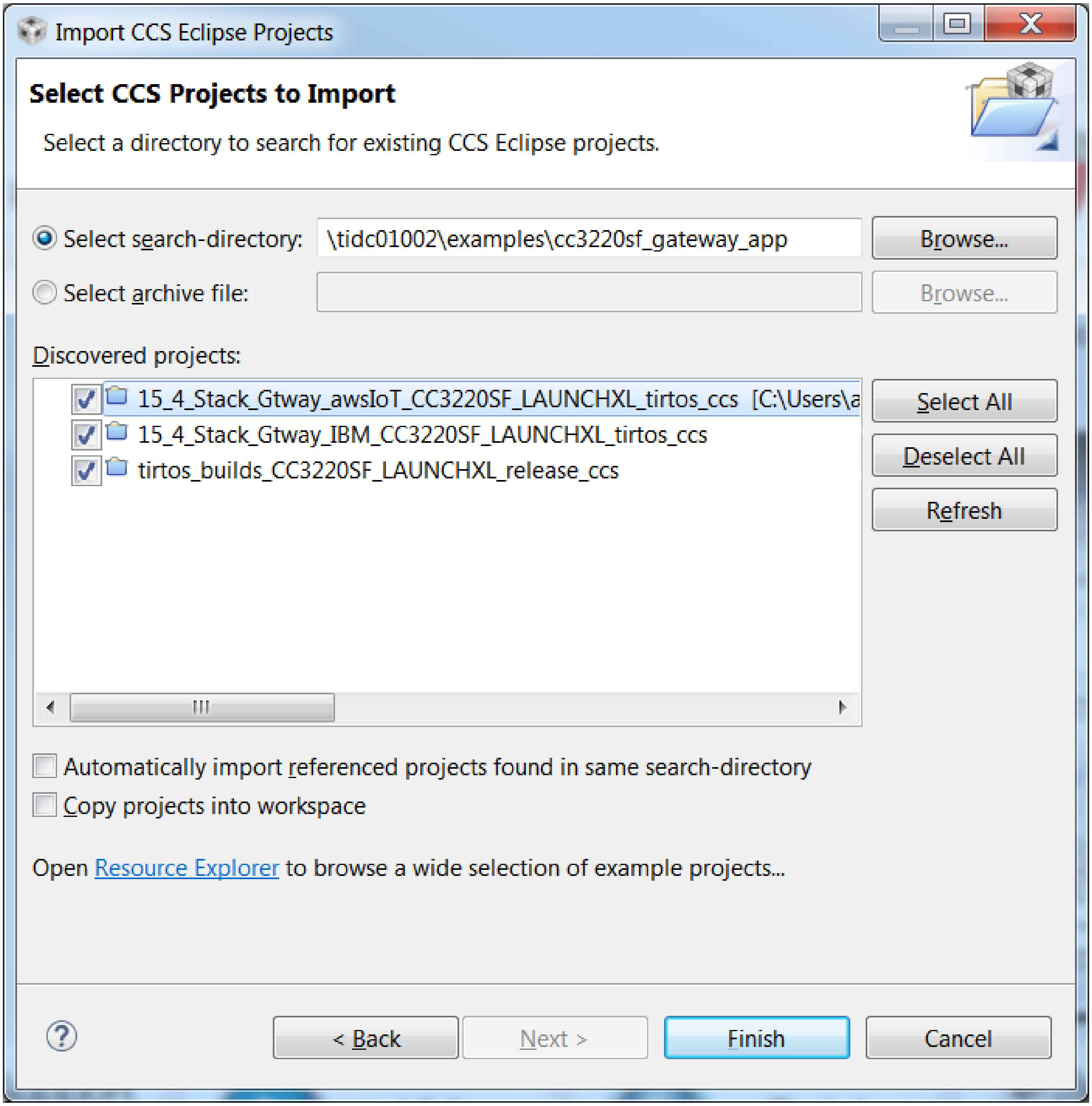
Imports existing CCS Eclipse projects into workspace.

Select an import wizard:

type filter text

Figure 13. Select Project to Import

4. Click on *Browse...* and navigate to *<S2C_Repo_Dir>\tidc01002\examples\cc3220sf_gateway_app*.



Import CCS Eclipse Projects

Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

● Select search-directory: \tidc01002\examples\cc3220sf_gateway_app

○ Select archive file:

Discovered projects:

☑ 15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs [C:\Users\a

☑ 15_4_Stack_Gtway_IBM_CC3220SF_LAUNCHXL_tirtos_ccs

☑ tirtos_builds_CC3220SF_LAUNCHXL_release_ccs

□ Automatically import referenced projects found in same search-directory

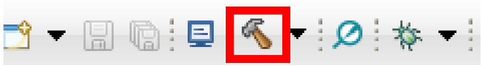
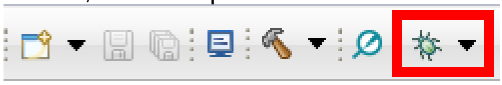
□ Copy projects into workspace

Open Resource Explorer to browse a wide selection of example projects...

Figure 14. Select Projects to Import

5. Select the desired examples to be imported and click on *Finish*.
6. *(AWS Example Only)* Request AWS IoT demo configuration from StackArmor by filling out the following form.
7. *(AWS Example Only)* Update the AWS configuration file:
   1. After receiving the certificates and configuration information from StackArmos open the aws_iot_config.h file, found in CloudService/AWS.
   2. Set the value of AWS_IOT_MQTT_HOST to the URL provided by StackArmor, it should look something like this "https://<random-string>.iot.us-east-1.amazonaws.com "

3. Modify AWS_IOT_MQTT_CLIENT_ID to a unique name for the device.
4. Replace the value of AWS_IOT_MY_THING_NAME with the extended address of the MAC-Cop LaunchPad.
5. Open the file certs.h, found in the CloudService/AWS directory.
6. Search for “USER STEP” and update the CA root certificate string, the client certificate string, and the client (private) key string.

8. Compile the example by clicking on the Build button ().
9. If the CC3220SF LaunchPad is already in developer mode, the example can be run and debugged directly from CCS by clicking on the debug button ().
10. If the CC3220SF Launchpad is not in developer mode, follow Section 3.2.1.1.4.4 to flash and run the example.

3.2.1.1.4.3 Compiling the Mac-CoP from Source

These are optional steps if there is a need to re-compile the Mac-CoP firmware for the gateway as opposed to using the prebuilt CoP firmware.

When re-compiling the Mac CoP firmware, the pin configuration for the UART needs to be modified since the default Mac CoP example in the CC13xx SDK uses IOID 3 and IOID 2 pins for UART and the gateway uses IOID 11 and IOID 9. See steps below to modify the pin configuration.

1. Import the **coprocessor** example from the CC13xx SDK to CCS
2. Open the file CC13X0_LAUNCHXL.h and modify the two lines shown below.

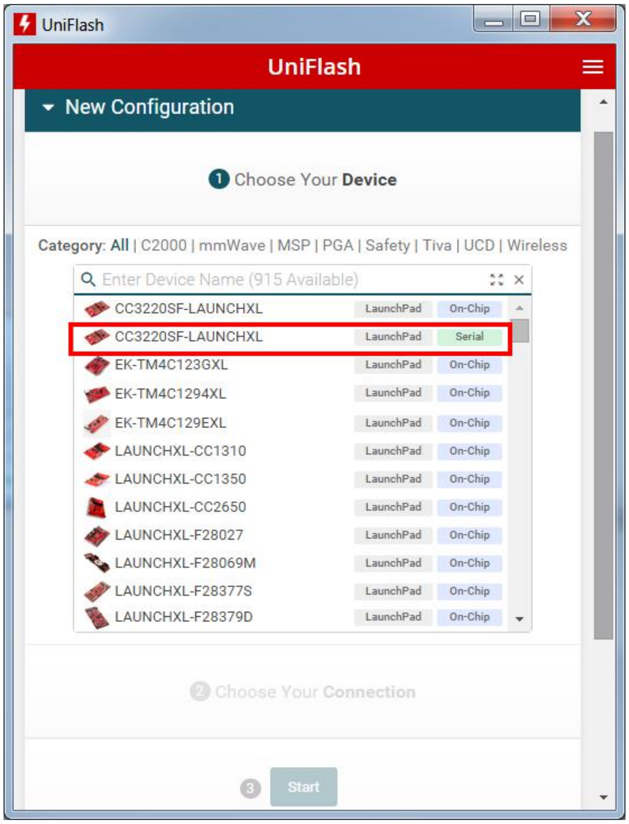
```
/* UART Board */
#define Board_UART_RX                IOID_9          /* RXD  */
#define Board_UART_TX                IOID_11         /* TXD  */
```

3. Now rebuild the example and flash it into the CC13xx LaunchPad

3.2.1.1.4.4 Creating an Image Creator Project in UniFlash

The following steps will allow the user to customize the example and use the new, updated files instead of the pre-build ones.

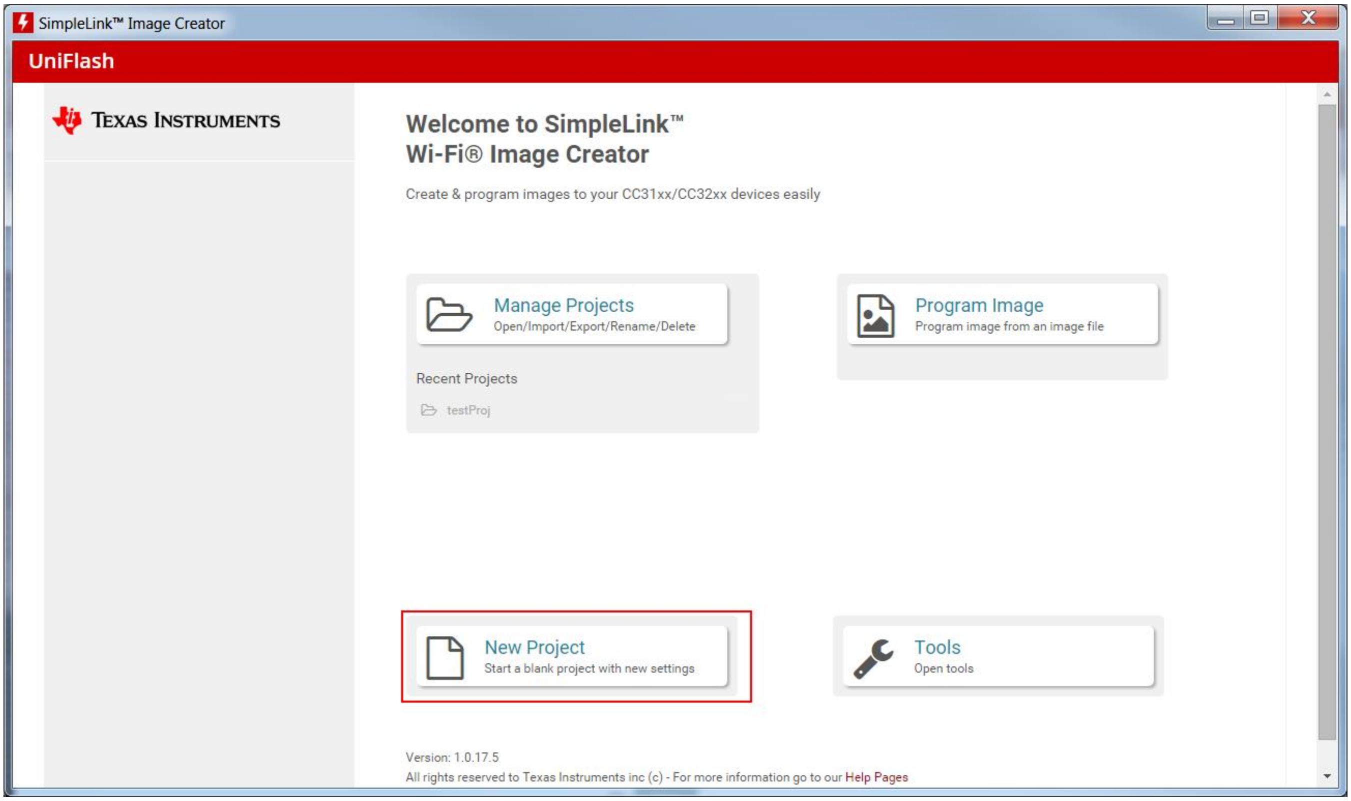
1. Open UniFlash.
2. On the *Choose your Device* section, select *CC3220SF-LAUNCHXL*. Make sure the *Serial* option is selected and not *On-Chip* as shown in Figure 15. Click on the *Start Image Creator* button.


UniFlash
UniFlash
New Configuration
1 Choose Your Device
Category: All | C2000 | mmWave | MSP | PGA | Safety | Tiva | UCD | Wireless
Enter Device Name (915 Available)
2 Choose Your Connection
3 Start

CC3220SF-LAUNCHXL	LaunchPad	On-Chip
CC3220SF-LAUNCHXL	LaunchPad	Serial
EK-TM4C123GXL	LaunchPad	On-Chip
EK-TM4C1294XL	LaunchPad	On-Chip
EK-TM4C129EXL	LaunchPad	On-Chip
LAUNCHXL-CC1310	LaunchPad	On-Chip
LAUNCHXL-CC1350	LaunchPad	On-Chip
LAUNCHXL-CC2650	LaunchPad	On-Chip
LAUNCHXL-F28027	LaunchPad	On-Chip
LAUNCHXL-F28069M	LaunchPad	On-Chip
LAUNCHXL-F28377S	LaunchPad	On-Chip
LAUNCHXL-F28379D	LaunchPad	On-Chip

Figure 15. Select CC3220 Device

3. After starting Image Creator, click on the *New Project* button as in Figure 16.



SimpleLink™ Image Creator

UniFlash

TEXAS INSTRUMENTS

Welcome to SimpleLink™
Wi-Fi® Image Creator

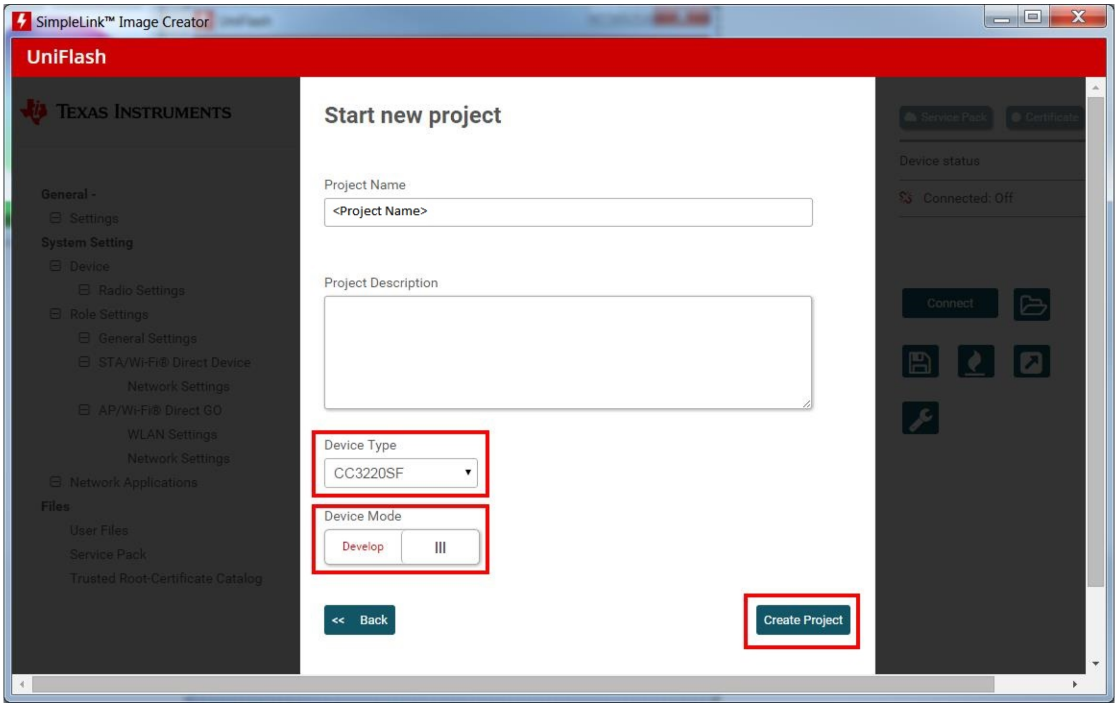
Create & program images to your CC31xx/CC32xx devices easily

Manage Projects
Open/Import/Export/Rename/Delete
Recent Projects
testProj
Program Image
Program image from an image file
New Project
Start a blank project with new settings
Tools
Open tools

Version: 1.0.17.5
All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 16. New Project

4. Enter a project name, select *CC3220SF* in the *Device Type* drop-down menu, make sure device mode is in *Develop*, and click on the *Create Project* button.



SimpleLink™ Image Creator

UniFlash

TEXAS INSTRUMENTS

Start new project

Project Name
<Project Name>

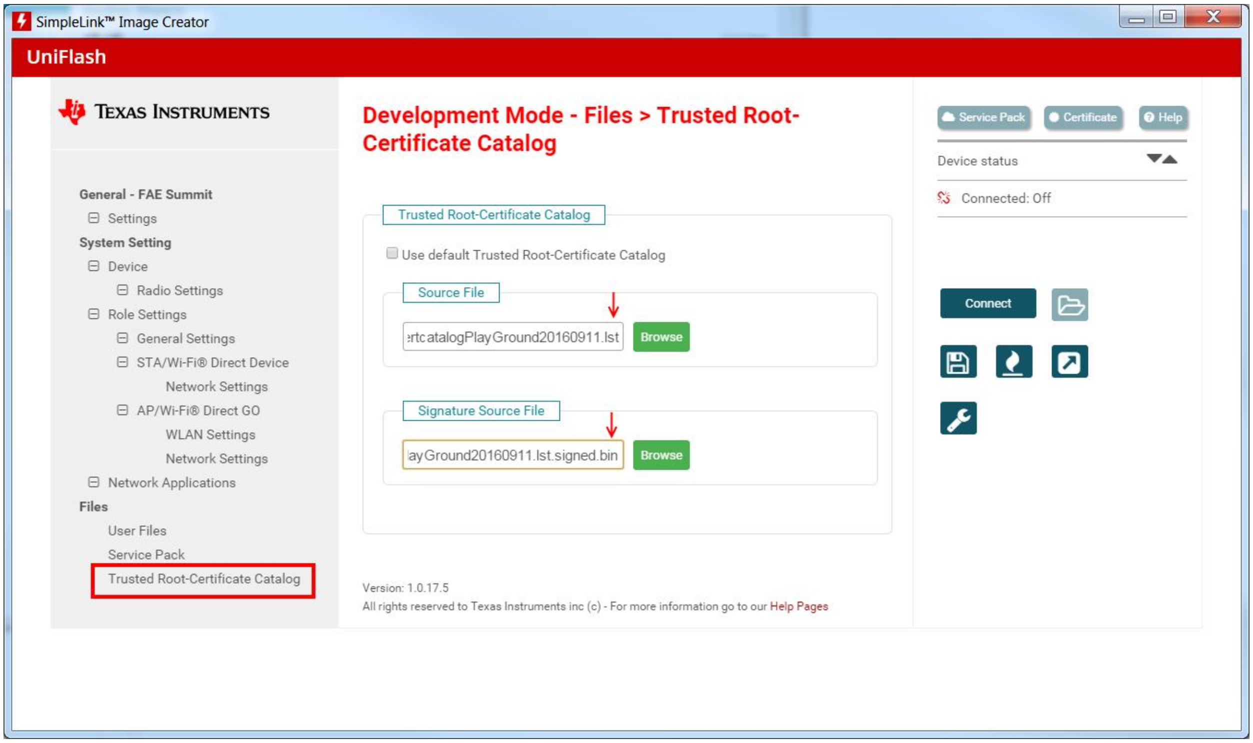
Project Description

Device Type
CC3220SF

Device Mode
Develop

Figure 17. Starting a New Project

5. Select *Trusted Root-Certificate Catalog* in the bottom-left corner and uncheck the *Use default Trusted Root-Certificate Catalog* box. Include the *Source File* (*certcatalogPlayGround20160911.lst*) and *Signature Source File* (*certcatalogPlayGround20160911.lst.signed.bin*) found in *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\certificate-catalog*.



Development Mode - Files > Trusted Root-Certificate Catalog

Trusted Root-Certificate Catalog

□ Use default Trusted Root-Certificate Catalog

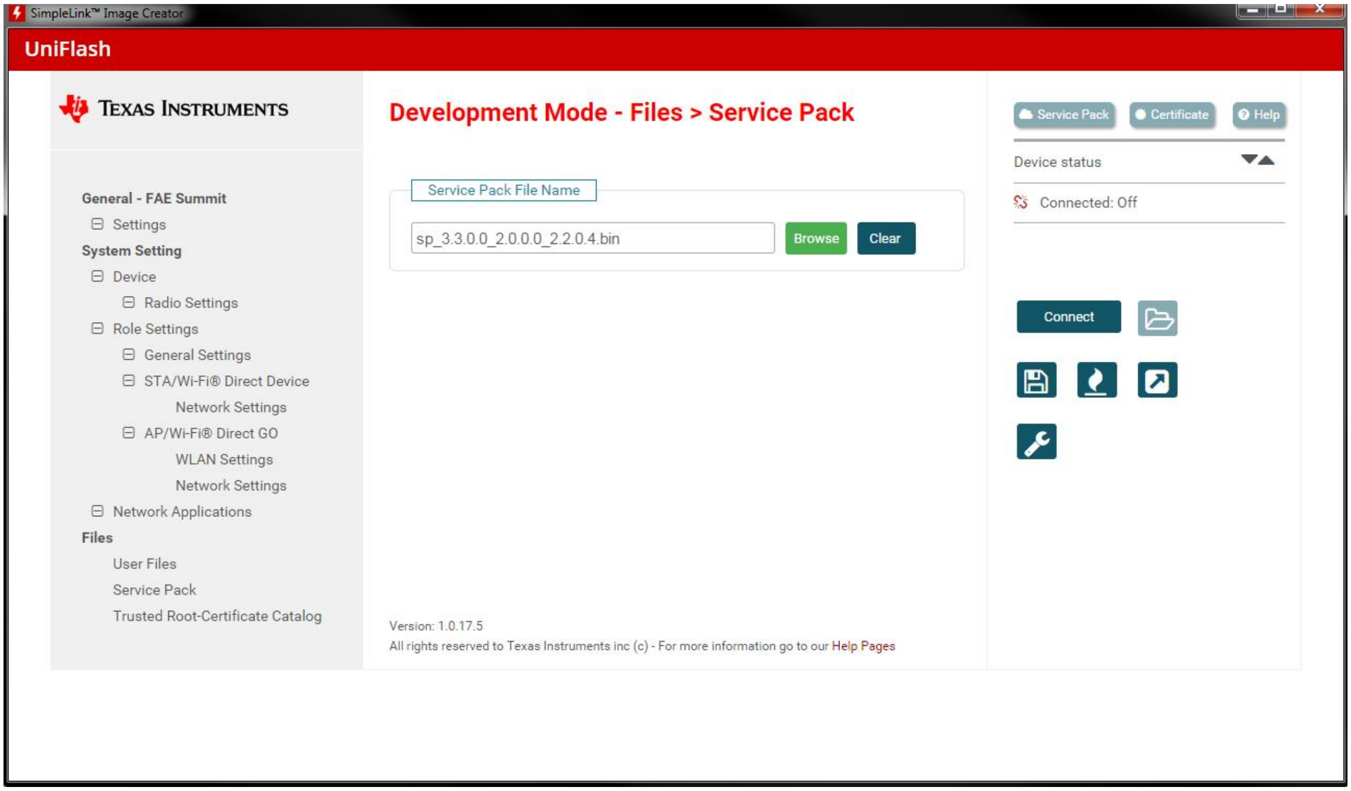
Source File: ertcatalogPlayGround20160911.lst

Signature Source File: ayGround20160911.lst.signed.bin

Version: 1.0.17.5
All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 18. Trusted Root Certificates

6. Select *Service Pack* in the bottom-left corner and include the service pack bin (*sp_3.3.0.0_2.0.0.0_2.2.0.4.bin*) found in *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\servicepack-cc3x20*.



Development Mode - Files > Service Pack

Service Pack File Name: sp_3.3.0.0_2.0.0.0_2.2.0.4.bin

Device status

Connected: Off

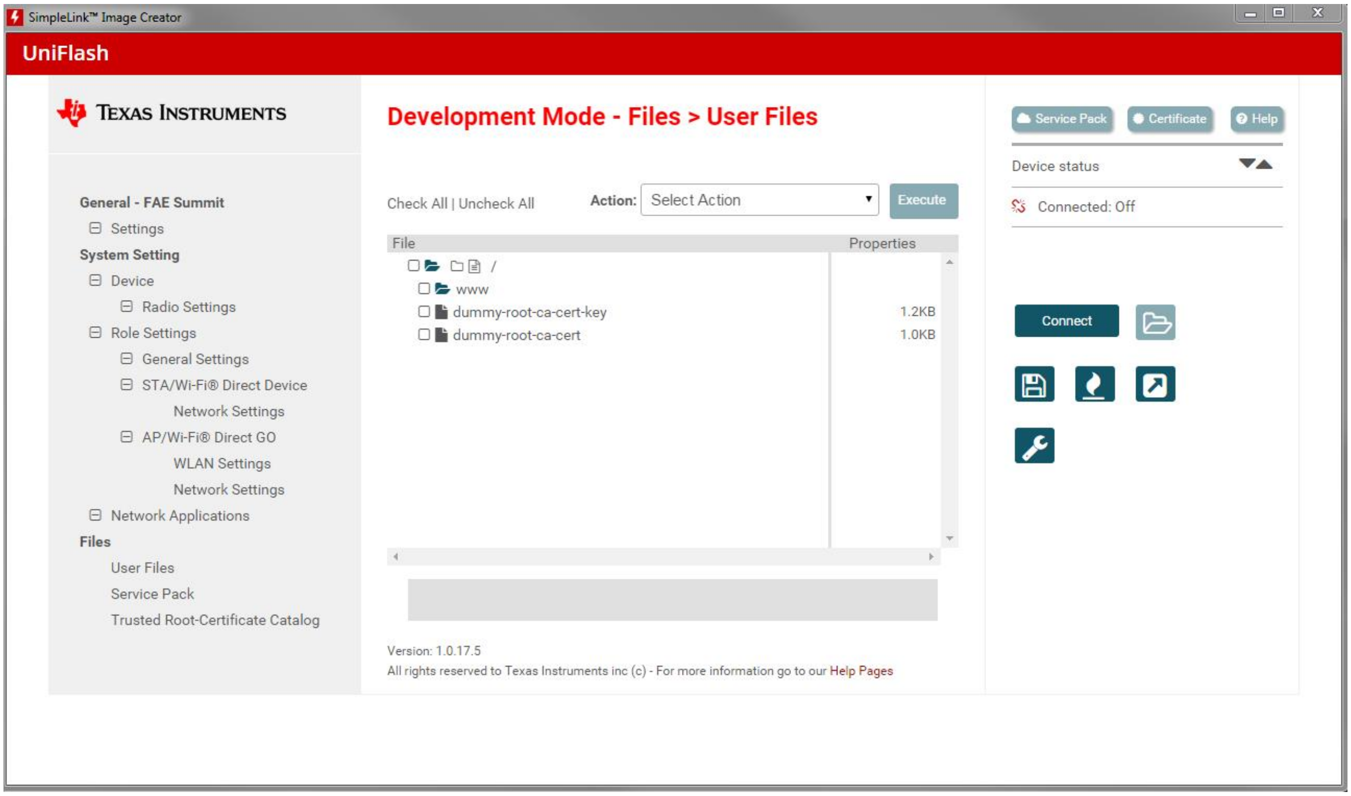
Version: 1.0.17.5

All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 19. Service Pack

7. Select *User Files* and include the *dummy-root-ca-cert* and *dummy-root-ca-cert-key* files by clicking on the *Add File* icon. These files can be found in *C:\ti\simplelink_cc32xx_sdk_1_50_00_06\tools\cc32xx_tools\certificate-playground.Create* a folder named *www* by clicking on the *New Folder* icon. This folder will contain all the web server required files.

NOTE: When adding the files do not select any of the options in the pop-up window—just click the *Write* button.



Development Mode - Files > User Files

File	Properties
□ /	
□ www	
□ dummy-root-ca-cert-key	1.2KB
□ dummy-root-ca-cert	1.0KB

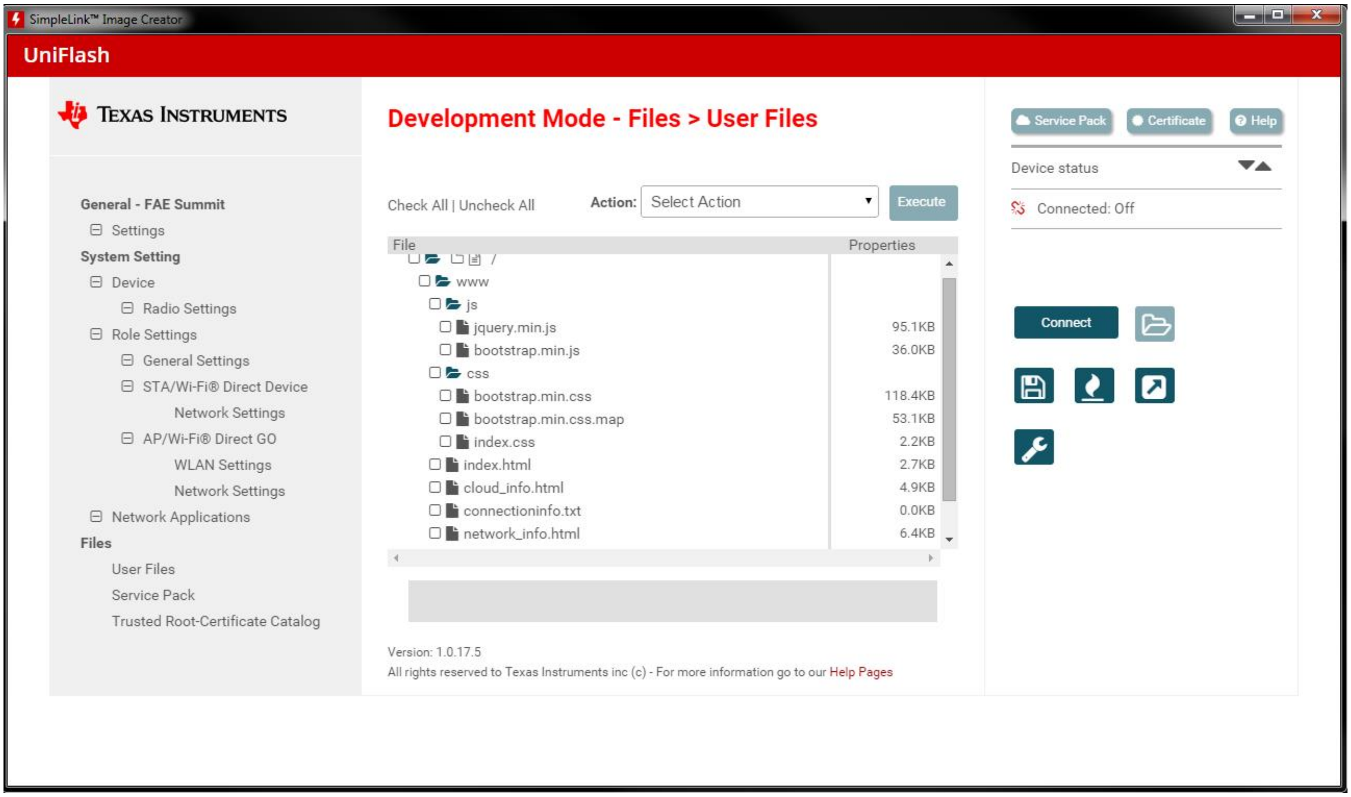
Version: 1.0.17.5

All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 20. User Files

8. Go to the following directory: *C:\<S2C Repo Directory>\tidc01002\src\www*, and recreate the same folder structure in the *www* folder in the devices *User Files*.

9. After creating all the folders in the www folder, transfer all the files in *C:\<S2C Repo Directory>\tidc01002\src\www* to the *www* folder of the device.



Development Mode - Files > User Files

Action: Select Action

Device status

Connected: Off

File	Properties
☐ /	
☐ www	
☐ js	
☐ jquery.min.js	95.1KB
☐ bootstrap.min.js	36.0KB
☐ css	
☐ bootstrap.min.css	118.4KB
☐ bootstrap.min.css.map	53.1KB
☐ index.css	2.2KB
☐ index.html	2.7KB
☐ cloud_info.html	4.9KB
☐ connectioninfo.txt	0.0KB
☐ network_info.html	6.4KB

Version: 1.0.17.5

All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 21. www Folder

10. On the drop-down box in the top-right corner, select *Select MCU Image*, and press *Browse*.

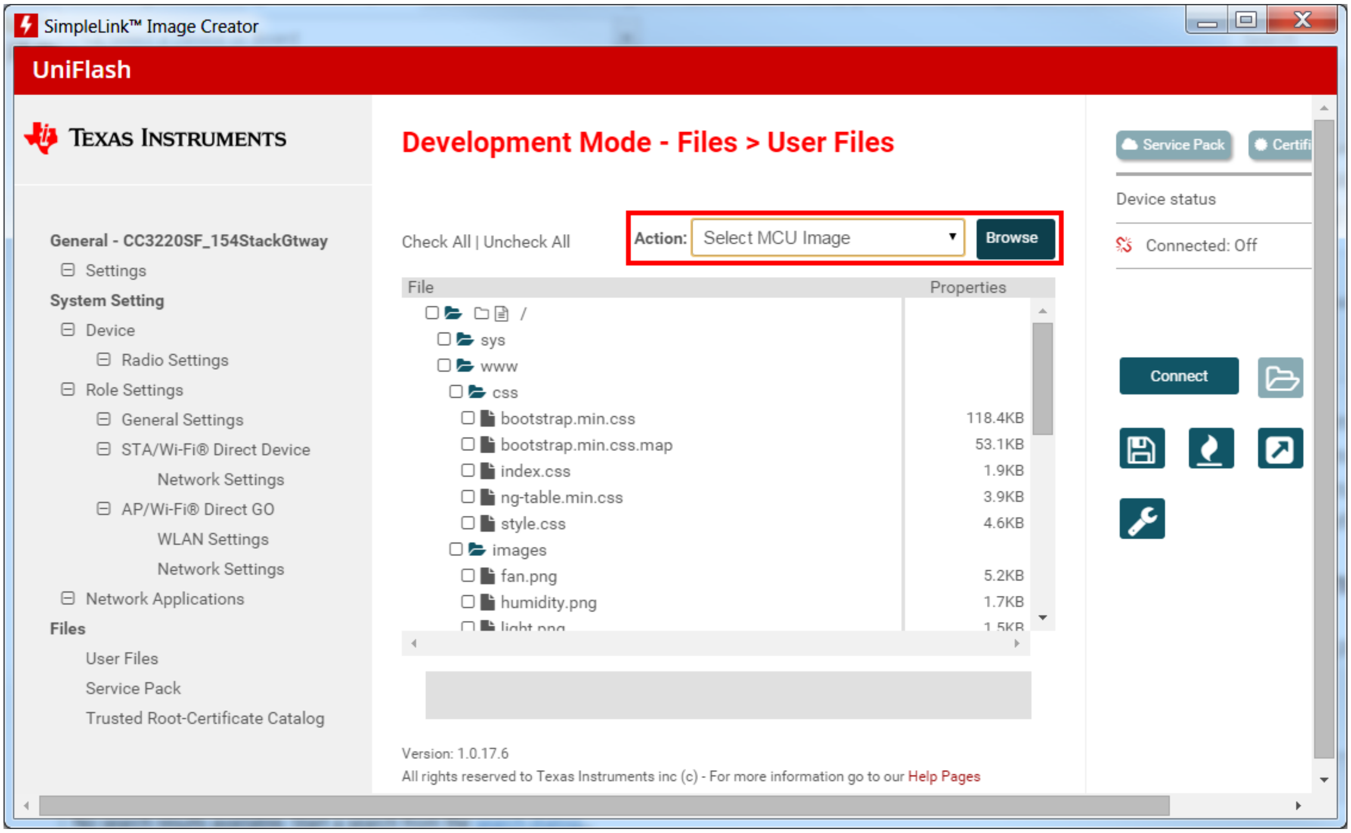
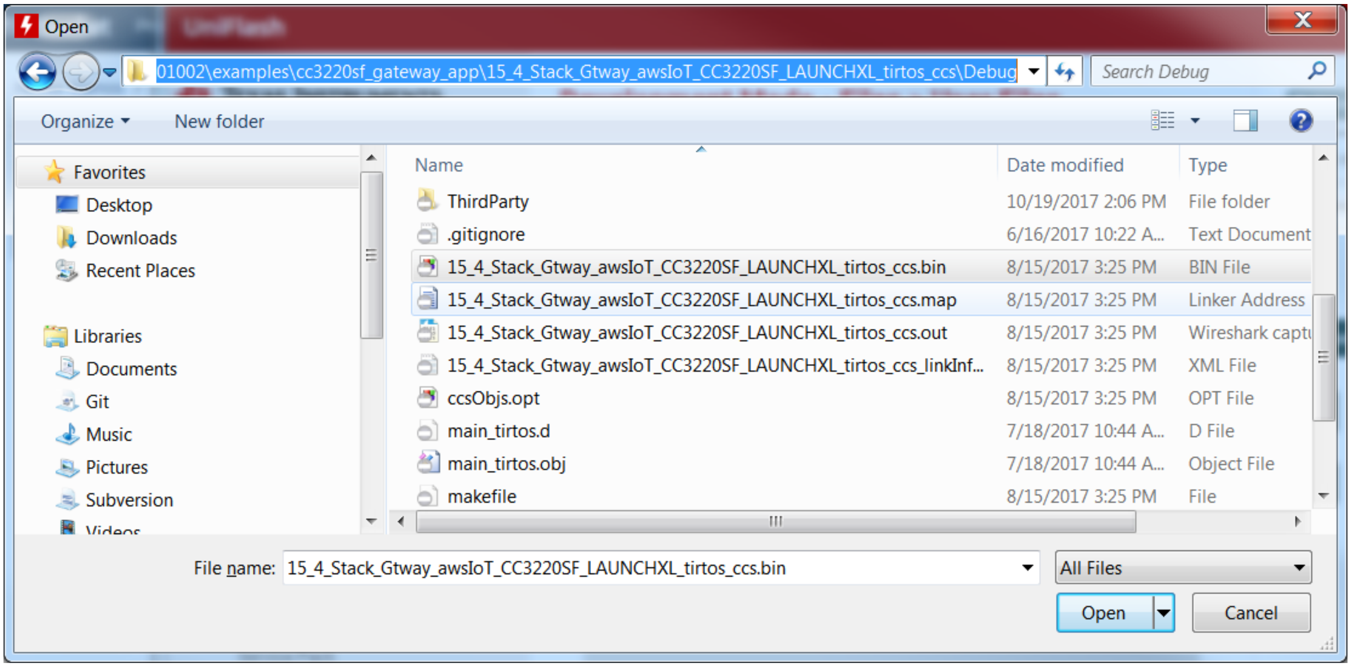


Figure 22. Select MCU Image

11. Navigate to the examples directory (*<S2C Repo Directory>\tidc01002\examples\cc3220sf_gateway_app*). Go into the directory of the example to be programmed in the CC3220sf and then go to the Debug folder. If the example has been successfully compiled in CCS then the Debug folder should have the files shown in Figure 23. Select the *.bin* file.



01002\examples\cc3220sf_gateway_app\15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs\Debug

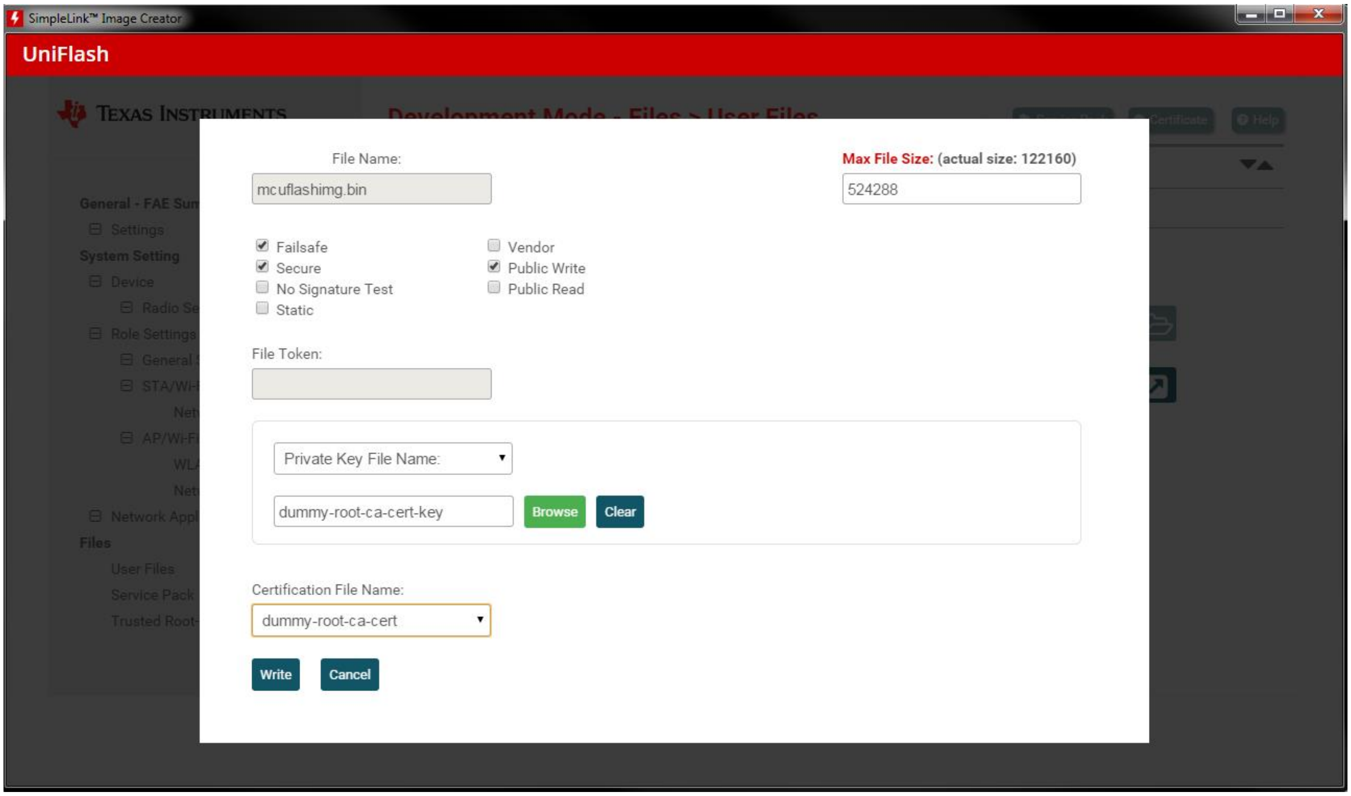
Name	Date modified	Type
ThirdParty	10/19/2017 2:06 PM	File folder
.gitignore	6/16/2017 10:22 A...	Text Document
15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs.bin	8/15/2017 3:25 PM	BIN File
15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs.map	8/15/2017 3:25 PM	Linker Address
15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs.out	8/15/2017 3:25 PM	Wireshark captu
15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs_linkInf...	8/15/2017 3:25 PM	XML File
ccsObjs.opt	8/15/2017 3:25 PM	OPT File
main_tirtos.d	7/18/2017 10:44 A...	D File
main_tirtos.obj	7/18/2017 10:44 A...	Object File
makefile	8/15/2017 3:25 PM	File

File name: 15_4_Stack_Gtway_awsIoT_CC3220SF_LAUNCHXL_tirtos_ccs.bin

All Files

Figure 23. Select Desired Example Binary

12. On the next menu select *Private Key Name:* and include the dummy-root-ca-cert-key. On the *Certification File Name:* select *dummy-root-ca-cert* from the list.



File Name: mcuflashimg.bin

Max File Size: (actual size: 122160) 524288

☑ Failsafe
☑ Secure
☐ No Signature Test
☐ Static
☐ Vendor
☑ Public Write
☐ Public Read

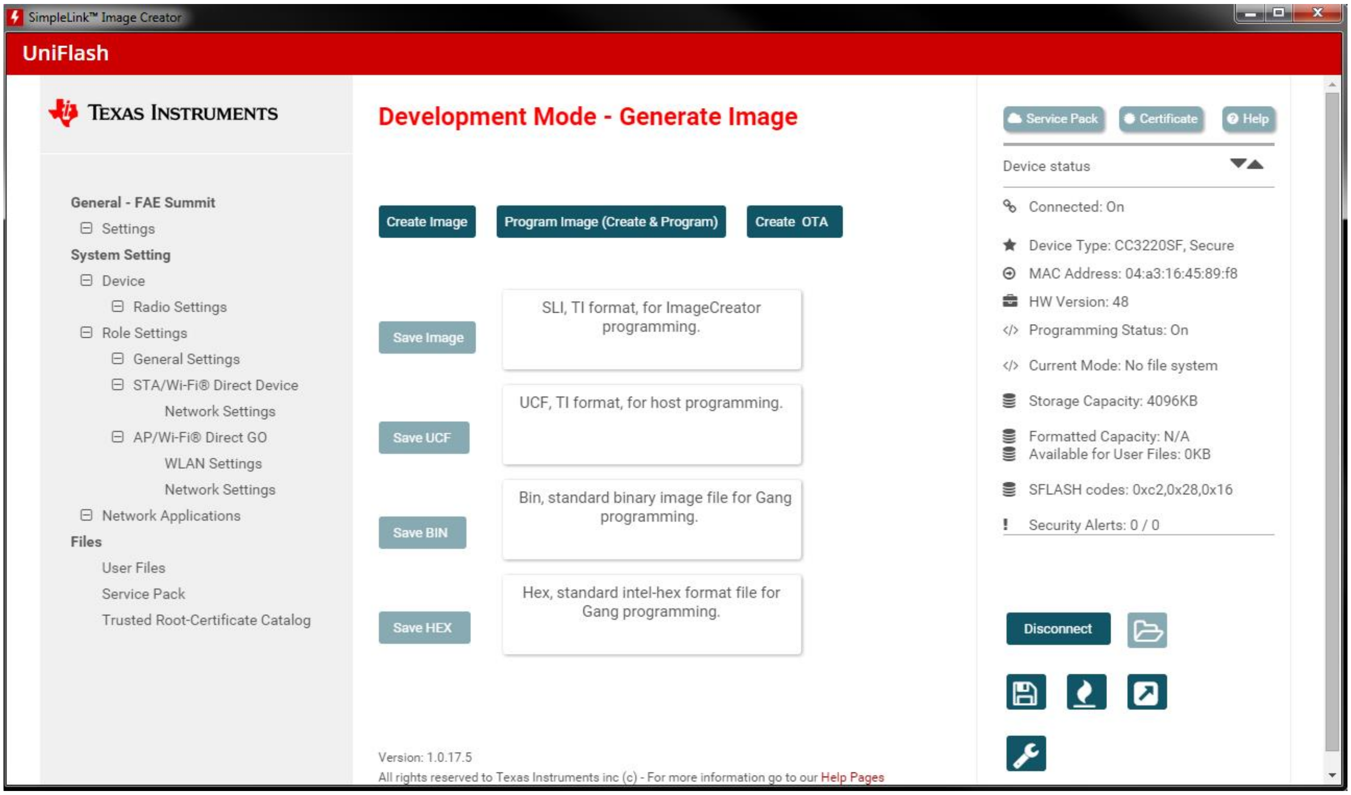
File Token:

Private Key File Name: dummy-root-ca-cert-key

Certification File Name: dummy-root-ca-cert

Figure 24. Select MCU Image

13. Connect the device to the PC through a USB cable, and press the *Connect* button found on the bottom-right corner. Once the device is connected, select the *Generate Image* button underneath the *Disconnect* button. Select *Program Image (Create & Program)*.



Development Mode - Generate Image

SLI, TI format, for ImageCreator programming.

UCF, TI format, for host programming.

Bin, standard binary image file for Gang programming.

Hex, standard intel-hex format file for Gang programming.

Device status

Connected: On

Device Type: CC3220SF, Secure

MAC Address: 04:a3:16:45:89:f8

HW Version: 48

Programming Status: On

Current Mode: No file system

Storage Capacity: 4096KB

Formatted Capacity: N/A
Available for User Files: 0KB

SFLASH codes: 0xc2,0x28,0x16

Security Alerts: 0 / 0

Version: 1.0.17.5
All rights reserved to Texas Instruments inc (c) - For more information go to our Help Pages

Figure 25. Generate Image

3.2.1.1.5 Connecting the MAC-CoP and the CC3220SF LaunchPad™

The following procedure applies for both CC13x0 and CC13x2.

1. Remove all jumpers at the center of the *MAC-Cop LaunchPad* except the *Reset* jumper highlighted in *green* . Also move the *VSENSE* jumperhighlighted in *blue* in Figure 26 to the Extern Pwr position.

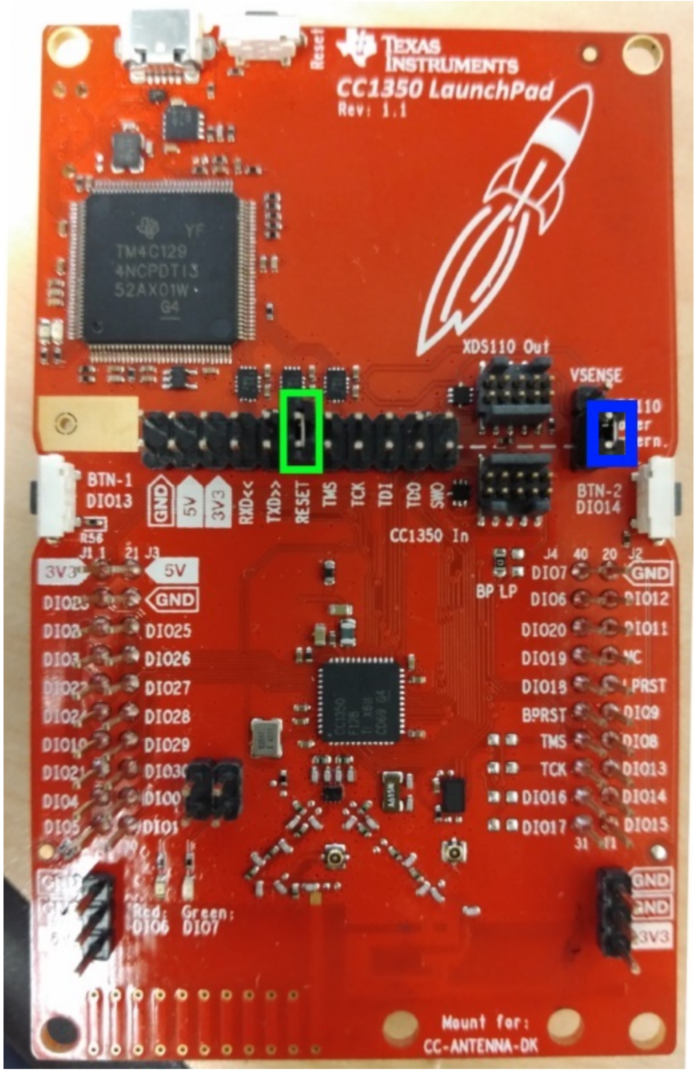


Figure 26. CC1350 LaunchPad™

2. Stack both LaunchPad development kits on top of each other as shown in Figure 27.

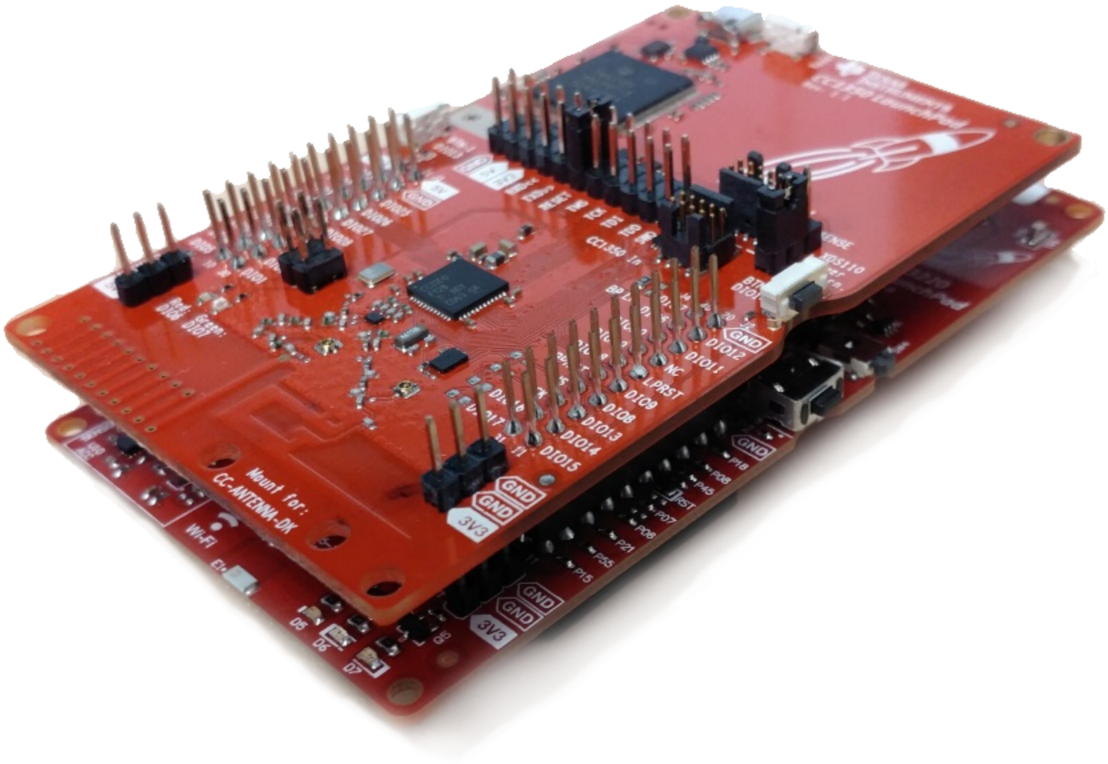


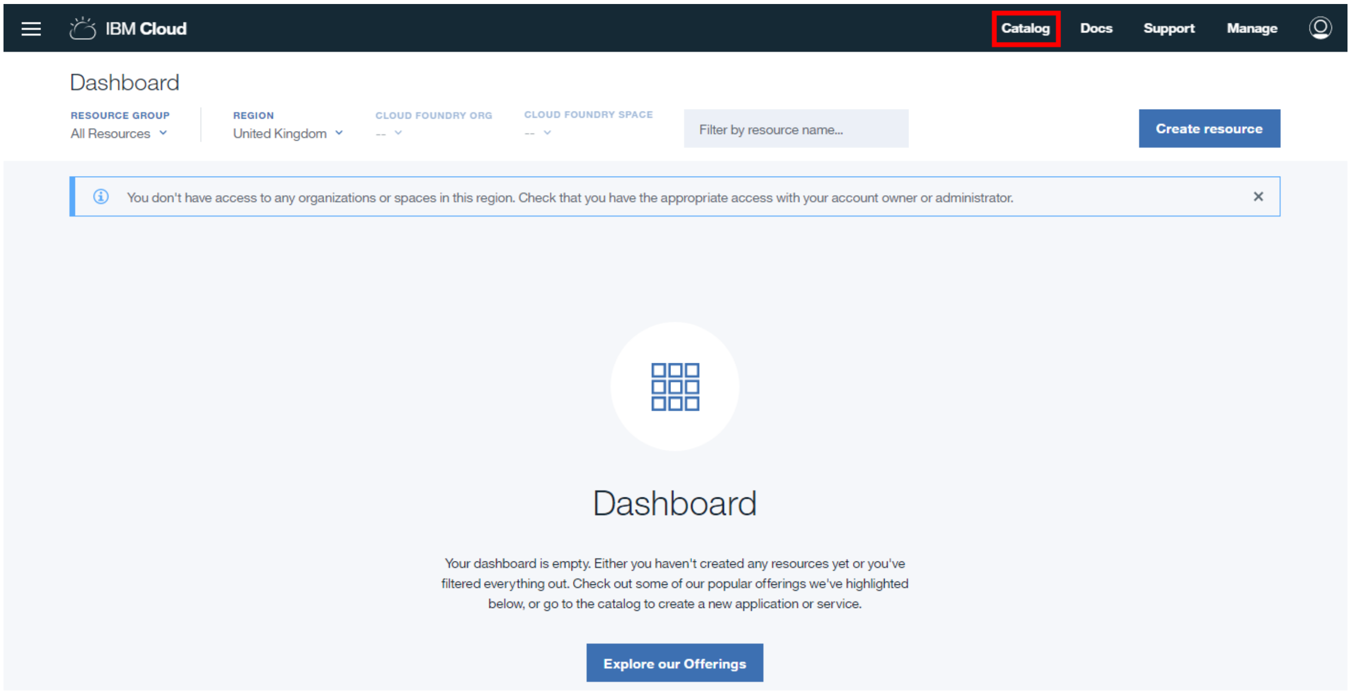
Figure 27. Stacked CC1350 and CC3220SF LaunchPad™ Development Kits

3. Connect a USB cable only to the CC3220 LaunchPad, and plug it in the PC.
4. Open a serial console (such as, PuTTy or Tera Term), select the COM port associated to the CC3220SF LaunchPad, and use the configuration below:

   Baud Rate: 115200
   Data: 8bit
   Parity: none
   Stop: 1bit
   Flow Control: none
5. Press the Reset button on the CC13x0 or CC13x2 LaunchPad attached to the CC3220SF, if the setup was successful, some debug logs should be displayed on the terminal..

3.2.1.1.6 Open and Configure an IBM Cloud™ Account

1. It is assumed that all the required software has already been installed. If not, install the required software now.
2. Go to IBM Cloud, click sign up if to create an account or log in if you already have an account.
3. Once logged in, click on the *Catalog* tab located on the top-right corner.



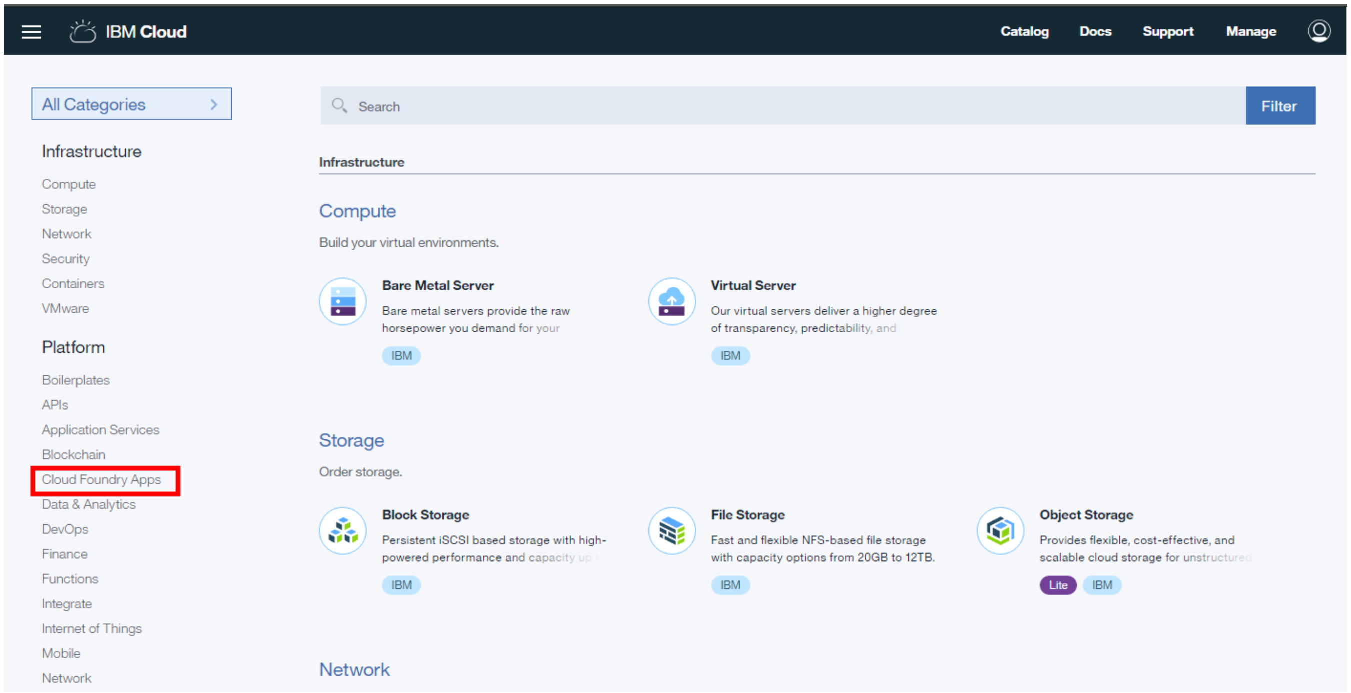
You don't have access to any organizations or spaces in this region. Check that you have the appropriate access with your account owner or administrator.

Dashboard

Your dashboard is empty. Either you haven't created any resources yet or you've filtered everything out. Check out some of our popular offerings we've highlighted below, or go to the catalog to create a new application or service.

Figure 28. IBM® Cloud

4. After selecting Catalog, click on *Cloud Foundry Apps* on the left menu.



IBM Cloud   Catalog   Docs   Support   Manage

All Categories >

Infrastructure
Compute
Storage
Network
Security
Containers
VMware

Platform
Boilerplates
APIs
Application Services
Blockchain
Cloud Foundry Apps
Data & Analytics
DevOps
Finance
Functions
Integrate
Internet of Things
Mobile
Network

Search   Filter

Infrastructure

Compute

Build your virtual environments.

Bare Metal Server
Bare metal servers provide the raw horsepower you demand for your
IBM

Virtual Server
Our virtual servers deliver a higher degree of transparency, predictability, and
IBM

Storage

Order storage.

Block Storage
Persistent iSCSI based storage with high-powered performance and capacity up t
IBM

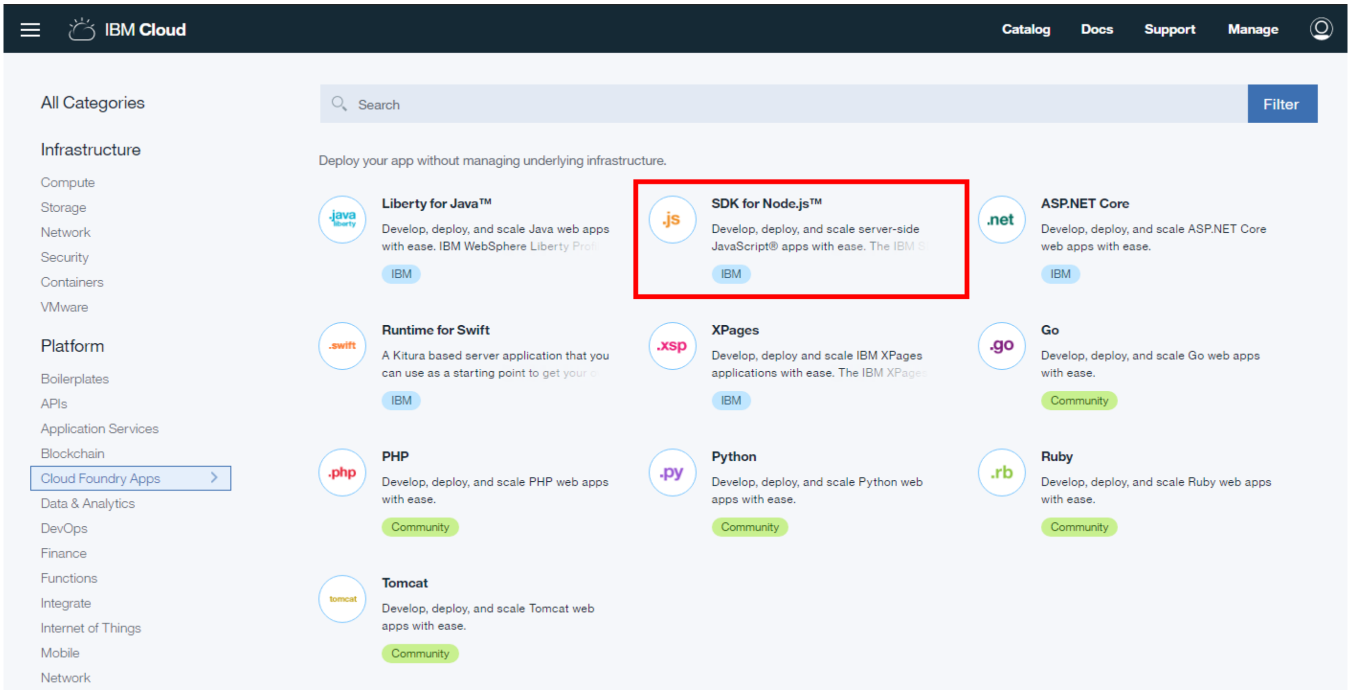
File Storage
Fast and flexible NFS-based file storage with capacity options from 20GB to 12TB.
IBM

Object Storage
Provides flexible, cost-effective, and scalable cloud storage for unstructured
Lite   IBM

Network

Figure 29. IBM Cloud™ Catalog

5. Choose *SDK for Node.js* from the options provided.



IBM Cloud   Catalog   Docs   Support   Manage

All Categories

Infrastructure
Compute
Storage
Network
Security
Containers
VMware

Platform
Boilerplates
APIs
Application Services
Blockchain
Cloud Foundry Apps >
Data & Analytics
DevOps
Finance
Functions
Integrate
Internet of Things
Mobile
Network

Search   Filter

Deploy your app without managing underlying infrastructure.

Liberty for Java™
Develop, deploy, and scale Java web apps with ease. IBM WebSphere Liberty Prof
IBM

SDK for Node.js™
Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM S
IBM

ASP.NET Core
Develop, deploy, and scale ASP.NET Core web apps with ease.
IBM

Runtime for Swift
A Kitura based server application that you can use as a starting point to get your o
IBM

XPages
Develop, deploy and scale IBM XPages applications with ease. The IBM XPages
IBM

Go
Develop, deploy, and scale Go web apps with ease.
Community

PHP
Develop, deploy, and scale PHP web apps with ease.
Community

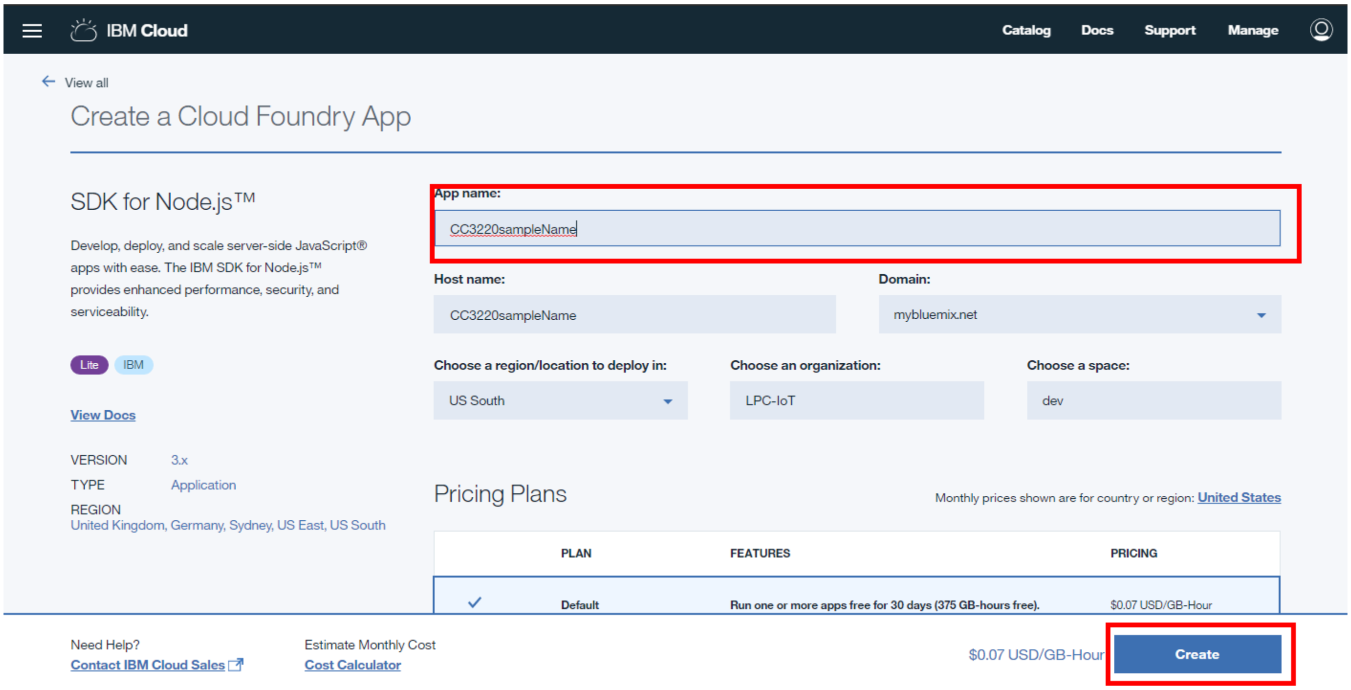
Python
Develop, deploy, and scale Python web apps with ease.
Community

Ruby
Develop, deploy, and scale Ruby web apps with ease.
Community

Tomcat
Develop, deploy, and scale Tomcat web apps with ease.
Community

Figure 30. Create Node.js Application

6. Enter a name for the new Node.js application under the *App name* text box, and press the *Create* button on the bottom-right corner.



Create a Cloud Foundry App

SDK for Node.js™

Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.

VERSION 3.x
TYPE Application
REGION
United Kingdom, Germany, Sydney, US East, US South

App name: CC3220sampleName

Host name: CC3220sampleName

Domain: mybluemix.net

Choose a region/location to deploy in: US South

Choose an organization: LPC-IoT

Choose a space: dev

Pricing Plans

Monthly prices shown are for country or region: United States

	PLAN	FEATURES	PRICING
✓	Default	Run one or more apps free for 30 days (375 GB-hours free).	$0.07 USD/GB-Hour

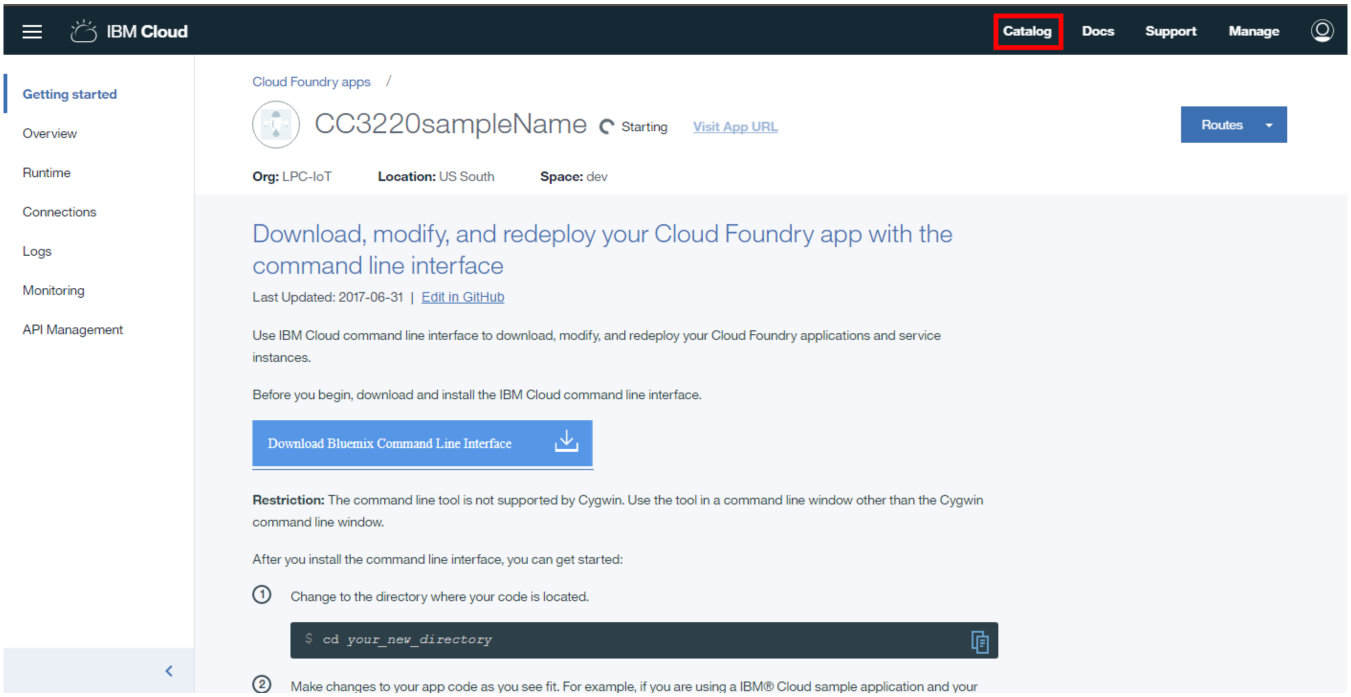
Need Help?
Contact IBM Cloud Sales

Estimate Monthly Cost
Cost Calculator

$0.07 USD/GB-Hour

Figure 31. Name and Create Application

7. Once the app has been created, click on the *Catalog*tab located on the top-right corner.



CC3220sampleName Starting

Org: LPC-IoT Location: US South Space: dev

Download, modify, and redeploy your Cloud Foundry app with the command line interface

Last Updated: 2017-06-31 | Edit in GitHub

Use IBM Cloud command line interface to download, modify, and redeploy your Cloud Foundry applications and service instances.

Before you begin, download and install the IBM Cloud command line interface.

Restriction: The command line tool is not supported by Cygwin. Use the tool in a command line window other than the Cygwin command line window.

After you install the command line interface, you can get started:

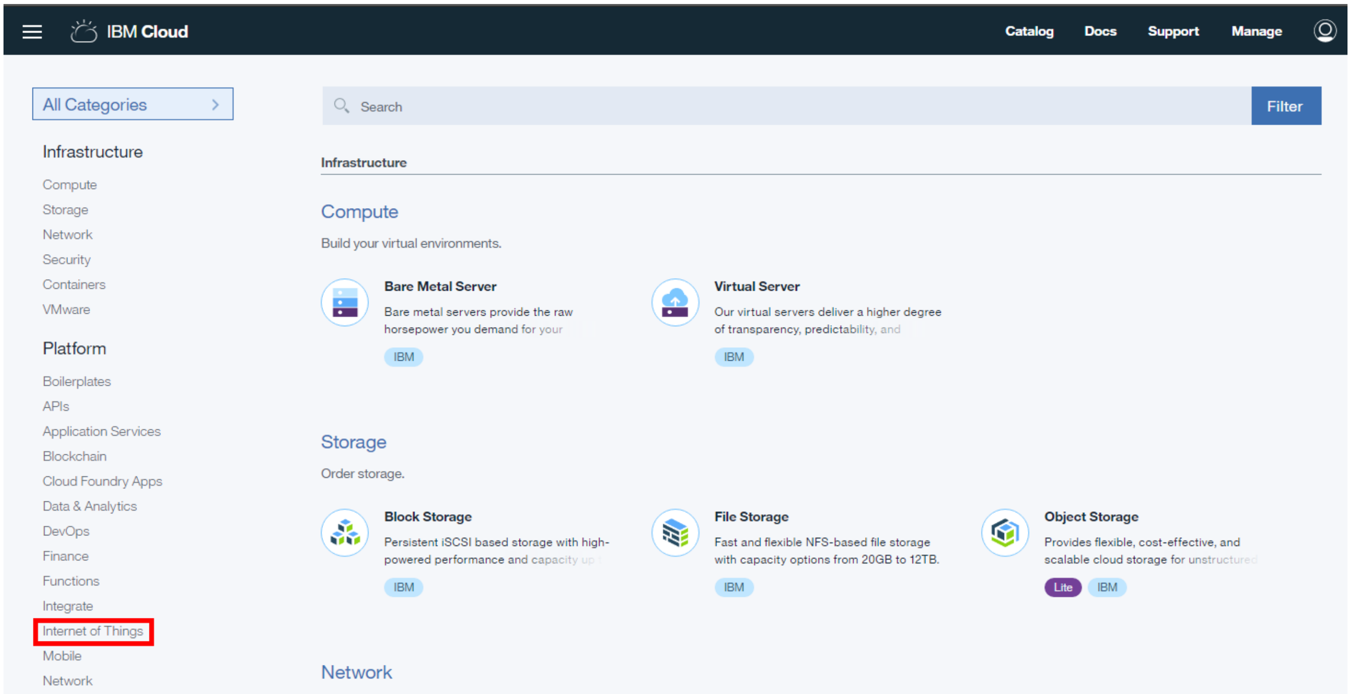
1. Change to the directory where your code is located.

```
$ cd your_new_directory
```

2. Make changes to your app code as you see fit. For example, if you are using a IBM® Cloud sample application and your

Figure 32. Go to Catalog

8. After selecting the Catalog, click on *Internet of Things* under *Services* on the left menu



Infrastructure

Compute

Build your virtual environments.

Bare Metal Server
Bare metal servers provide the raw horsepower you demand for your
IBM

Virtual Server
Our virtual servers deliver a higher degree of transparency, predictability, and
IBM

Storage

Order storage.

Block Storage
Persistent iSCSI based storage with high-powered performance and capacity up
IBM

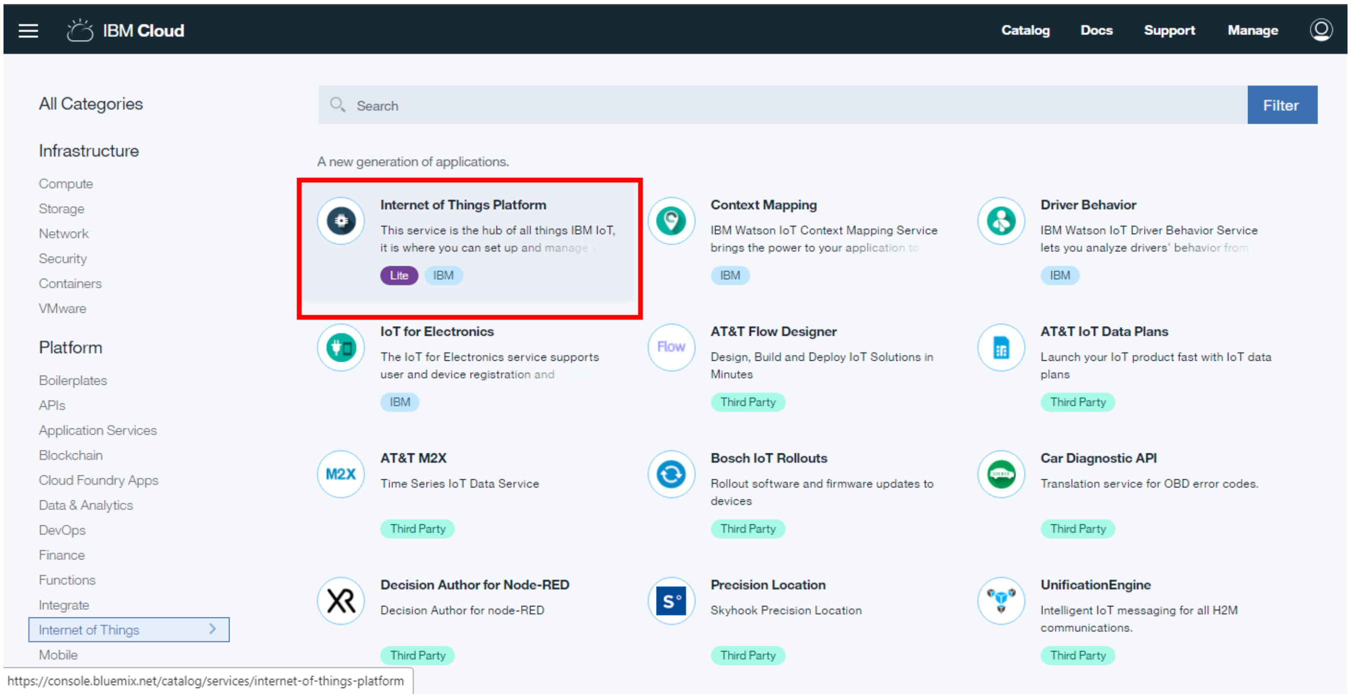
File Storage
Fast and flexible NFS-based file storage with capacity options from 20GB to 12TB.
IBM

Object Storage
Provides flexible, cost-effective, and scalable cloud storage for unstructured
Lite IBM

Network

Figure 33. IBM Cloud™ Catalog

9. Choose *Internet of Things Platform* from the options provided.



A new generation of applications.

Internet of Things Platform
This service is the hub of all things IBM IoT, it is where you can set up and manage
Lite IBM

Context Mapping
IBM Watson IoT Context Mapping Service brings the power to your application to
IBM

Driver Behavior
IBM Watson IoT Driver Behavior Service lets you analyze drivers' behavior from
IBM

IoT for Electronics
The IoT for Electronics service supports user and device registration and
IBM

AT&T Flow Designer
Design, Build and Deploy IoT Solutions in Minutes
Third Party

AT&T IoT Data Plans
Launch your IoT product fast with IoT data plans
Third Party

AT&T M2X
Time Series IoT Data Service
Third Party

Bosch IoT Rollouts
Rollout software and firmware updates to devices
Third Party

Car Diagnostic API
Translation service for OBD error codes.
Third Party

Decision Author for Node-RED
Decision Author for node-RED
Third Party

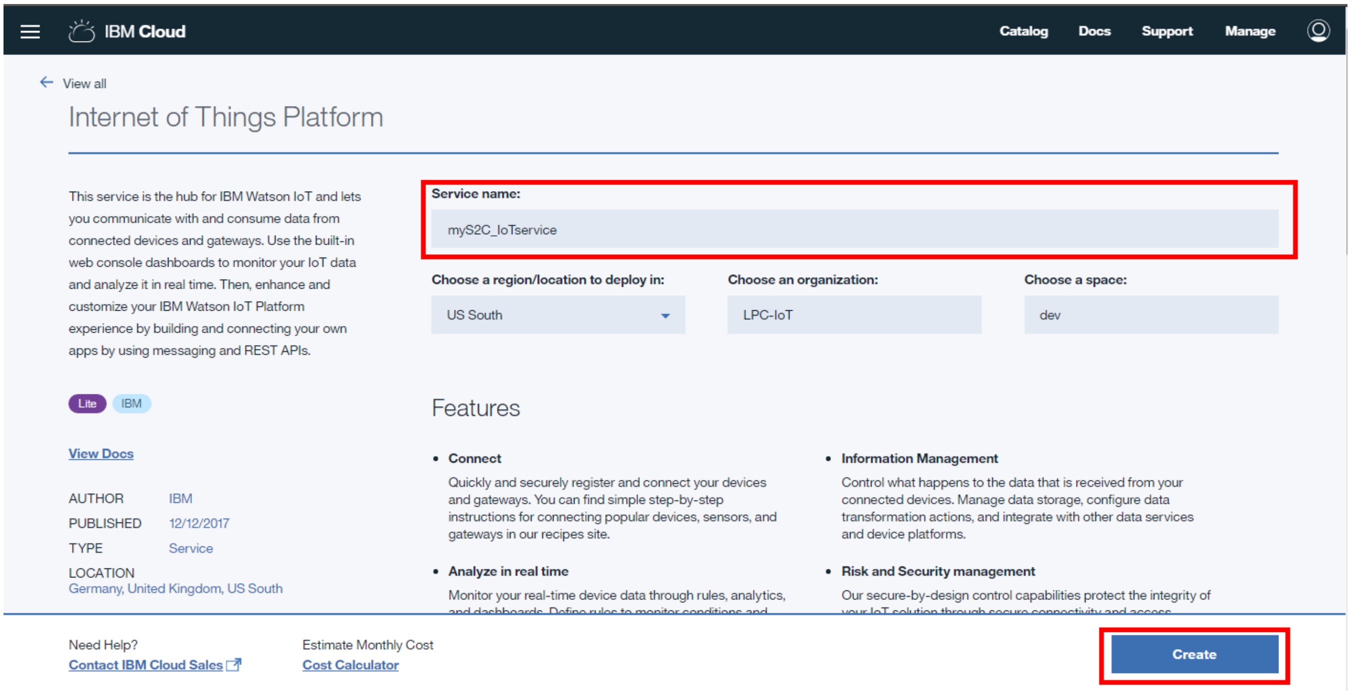
Precision Location
Skyhook Precision Location
Third Party

UnificationEngine
Intelligent IoT messaging for all H2M communications.
Third Party

https://console.bluemix.net/catalog/services/internet-of-things-platform

Figure 34. IoT Service Options

10. Give the new platform a service name. Click the *Create* button on the bottom-right corner.



Internet of Things Platform

This service is the hub for IBM Watson IoT and lets you communicate with and consume data from connected devices and gateways. Use the built-in web console dashboards to monitor your IoT data and analyze it in real time. Then, enhance and customize your IBM Watson IoT Platform experience by building and connecting your own apps by using messaging and REST APIs.

AUTHOR	IBM
PUBLISHED	12/12/2017
TYPE	Service
LOCATION	Germany, United Kingdom, US South

Service name: myS2C_IoTservice

Choose a region/location to deploy in: US South

Choose an organization: LPC-IoT

Choose a space: dev

Features

- **Connect**
  Quickly and securely register and connect your devices and gateways. You can find simple step-by-step instructions for connecting popular devices, sensors, and gateways in our recipes site.
- **Analyze in real time**
  Monitor your real-time device data through rules, analytics,
- **Information Management**
  Control what happens to the data that is received from your connected devices. Manage data storage, configure data transformation actions, and integrate with other data services and device platforms.
- **Risk and Security management**
  Our secure-by-design control capabilities protect the integrity of

Figure 35. Create IoT Platform

11. Click on the Launch button as shown below.

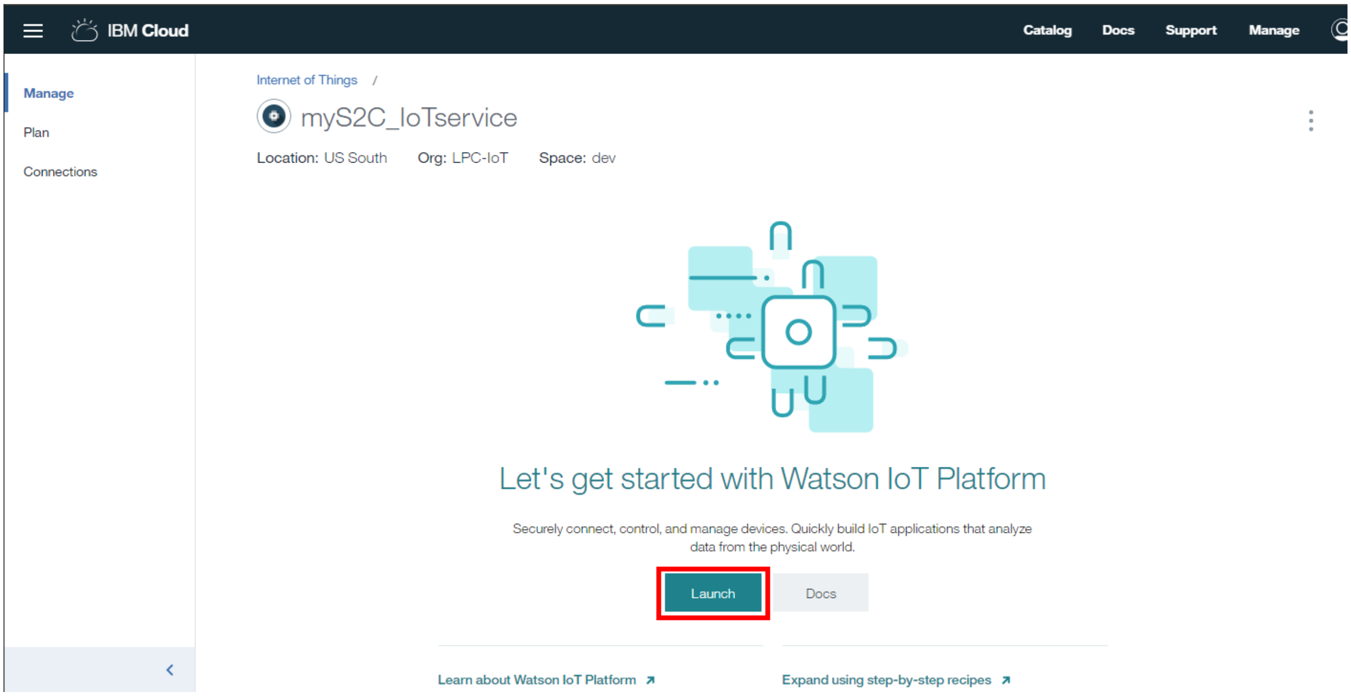

IBM Cloud
Catalog
Docs
Support
Manage
Manage
Plan
Connections
Internet of Things /
myS2C_IoTservice
Location: US South
Org: LPC-IoT
Space: dev
Let's get started with Watson IoT Platform
Securely connect, control, and manage devices. Quickly build IoT applications that analyze data from the physical world.
Launch
Docs
Learn about Watson IoT Platform
Expand using step-by-step recipes

Figure 36. Launch Service Screen

3.2.1.1.7 *Set Up Watson IoT™ Platform Service*

1. Once on the IBM Watson IoT Platform, go to Devices on the navigation bar on the left as shown in Figure 37.

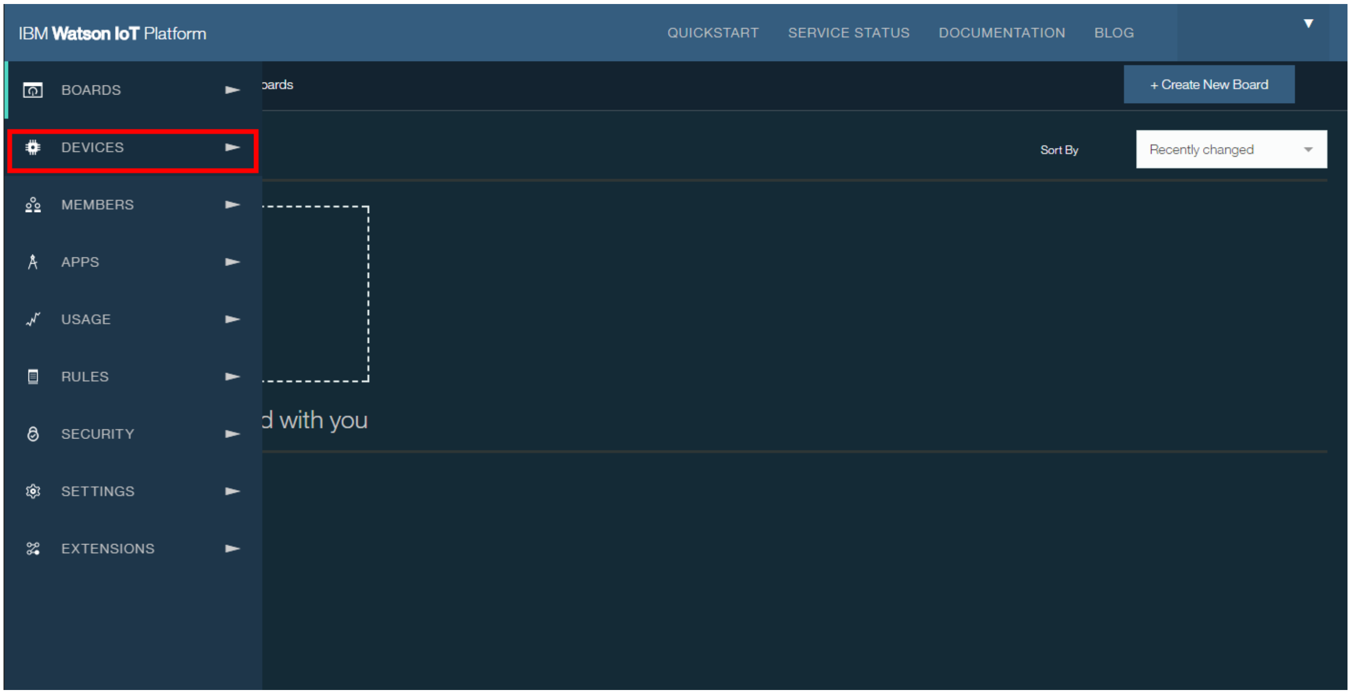
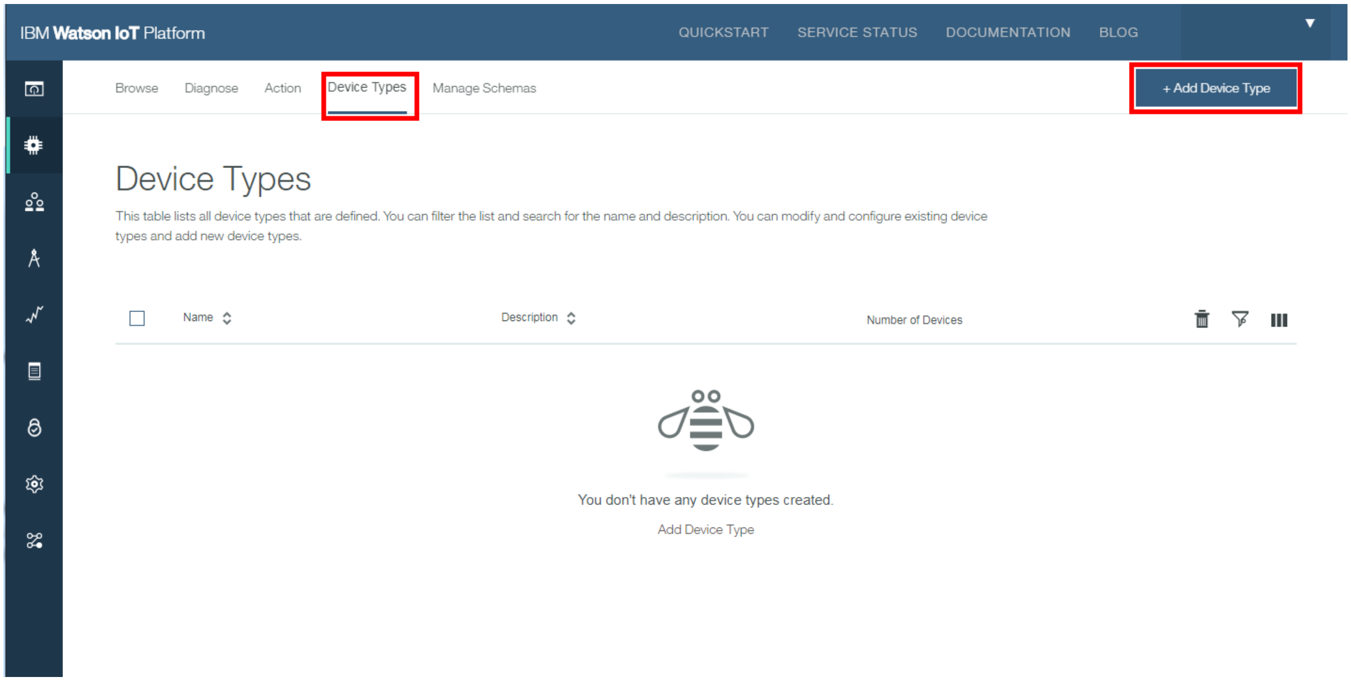


Figure 37. Go to Dashboard

2. Go to *Device Types* and Select *Add Device Type* on the top-right corner.



IBM Watson IoT Platform

QUICKSTART SERVICE STATUS DOCUMENTATION BLOG

Browse Diagnose Action Device Types Manage Schemas

+ Add Device Type

Device Types

This table lists all device types that are defined. You can filter the list and search for the name and description. You can modify and configure existing device types and add new device types.

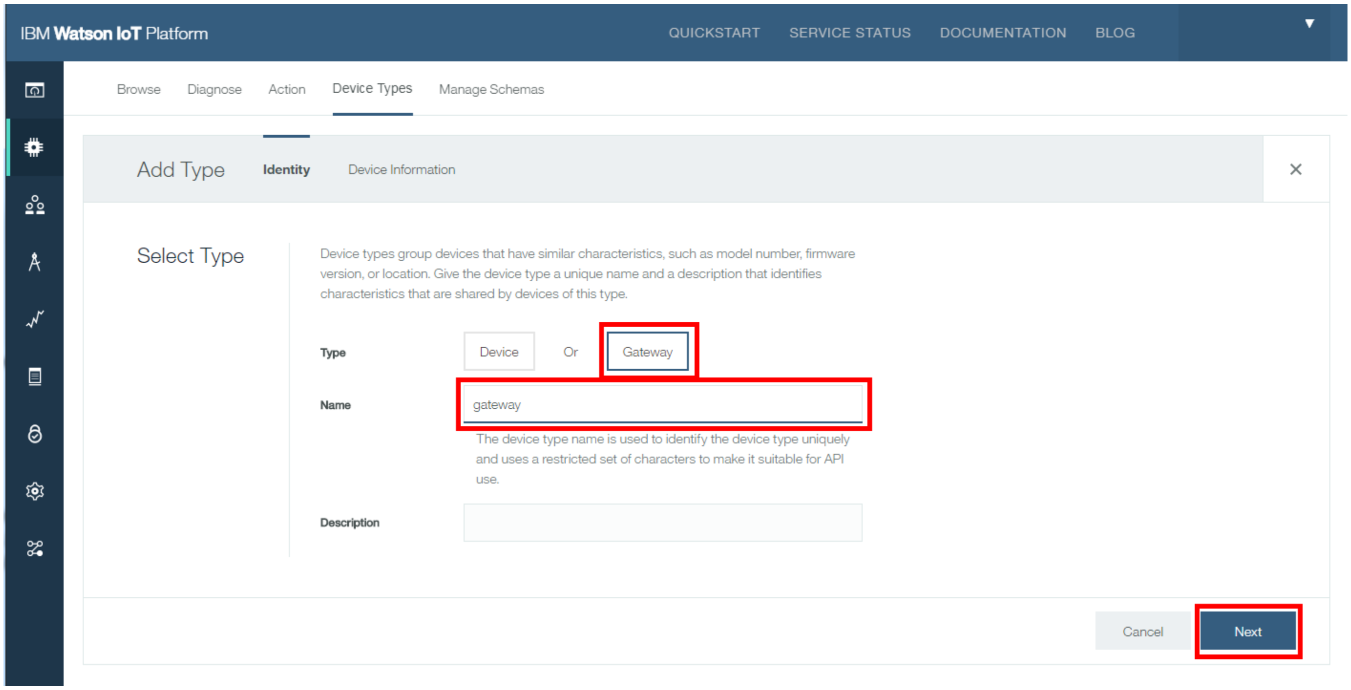
□	Name	Description	Number of Devices

You don't have any device types created.

Add Device Type

Figure 38. Add Device Type

3. Select *Gateway under Type and enter "gateway" for the name and click on Next* .



Device types group devices that have similar characteristics, such as model number, firmware version, or location. Give the device type a unique name and a description that identifies characteristics that are shared by devices of this type.

Type: Gateway

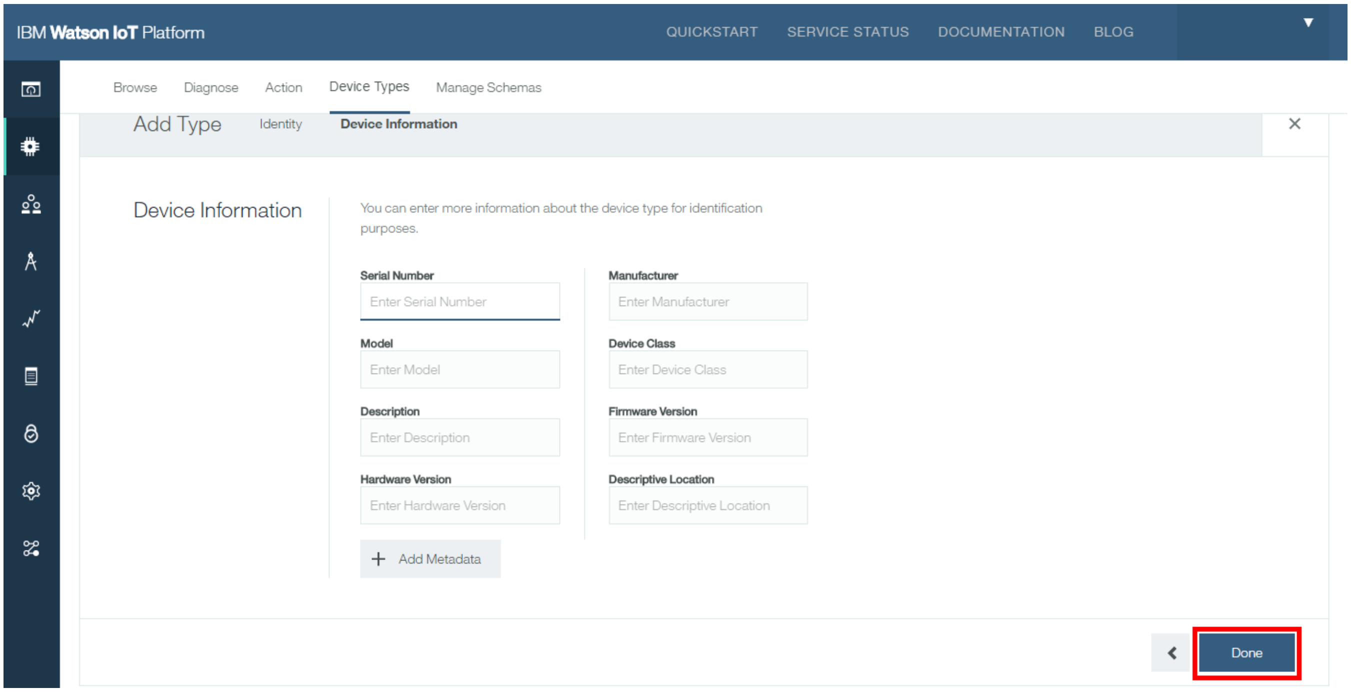
Name: gateway

The device type name is used to identify the device type uniquely and uses a restricted set of characters to make it suitable for API use.

Description:

Figure 39. Create Gateway Type

4. Click on Done.



You can enter more information about the device type for identification purposes.

Serial Number: Enter Serial Number

Manufacturer: Enter Manufacturer

Model: Enter Model

Device Class: Enter Device Class

Description: Enter Description

Firmware Version: Enter Firmware Version

Hardware Version: Enter Hardware Version

Descriptive Location: Enter Descriptive Location

Figure 40. Gateway Type Device Information

5. Go to the Browse tab and click on Add Device.

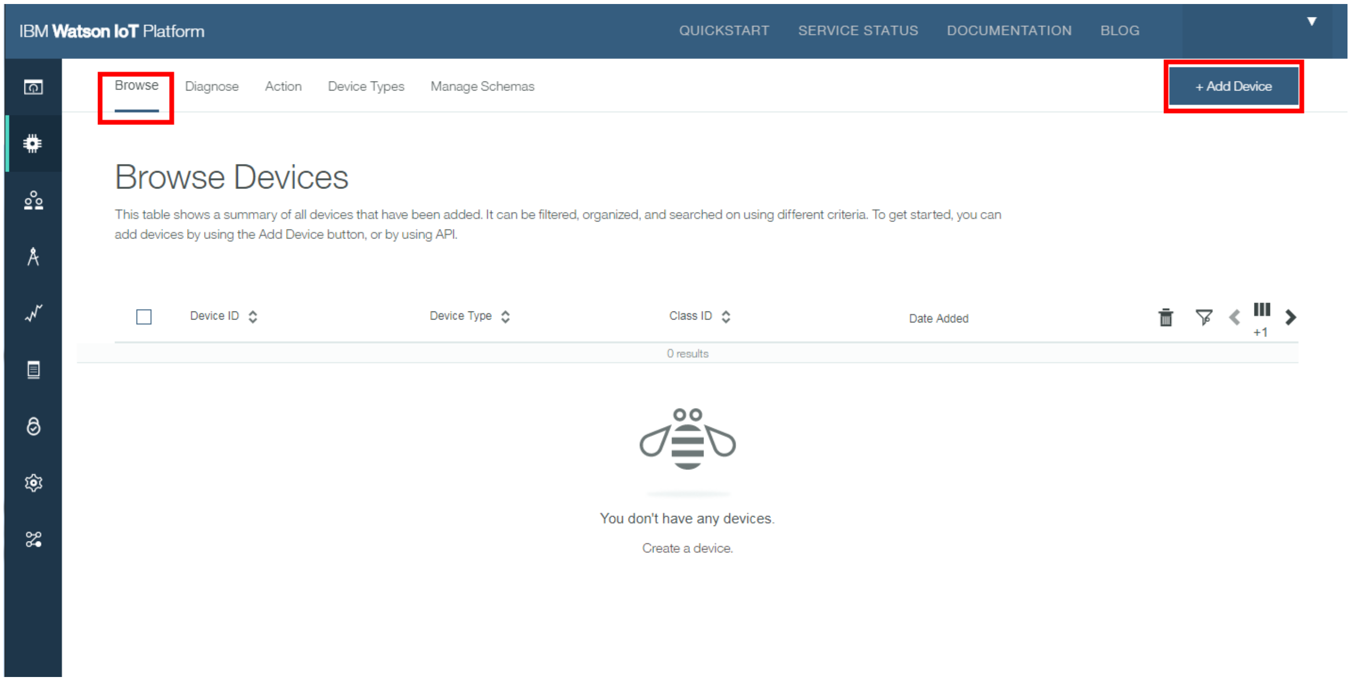

IBM Watson IoT Platform
QUICKSTART
SERVICE STATUS
DOCUMENTATION
BLOG
Browse
Diagnose
Action
Device Types
Manage Schemas
+ Add Device
Browse Devices
This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.
Device ID
Device Type
Class ID
Date Added
+1
0 results
You don't have any devices.
Create a device.

Figure 41. Browsing Devices

6. Select gateway as the device type and enter a device ID.

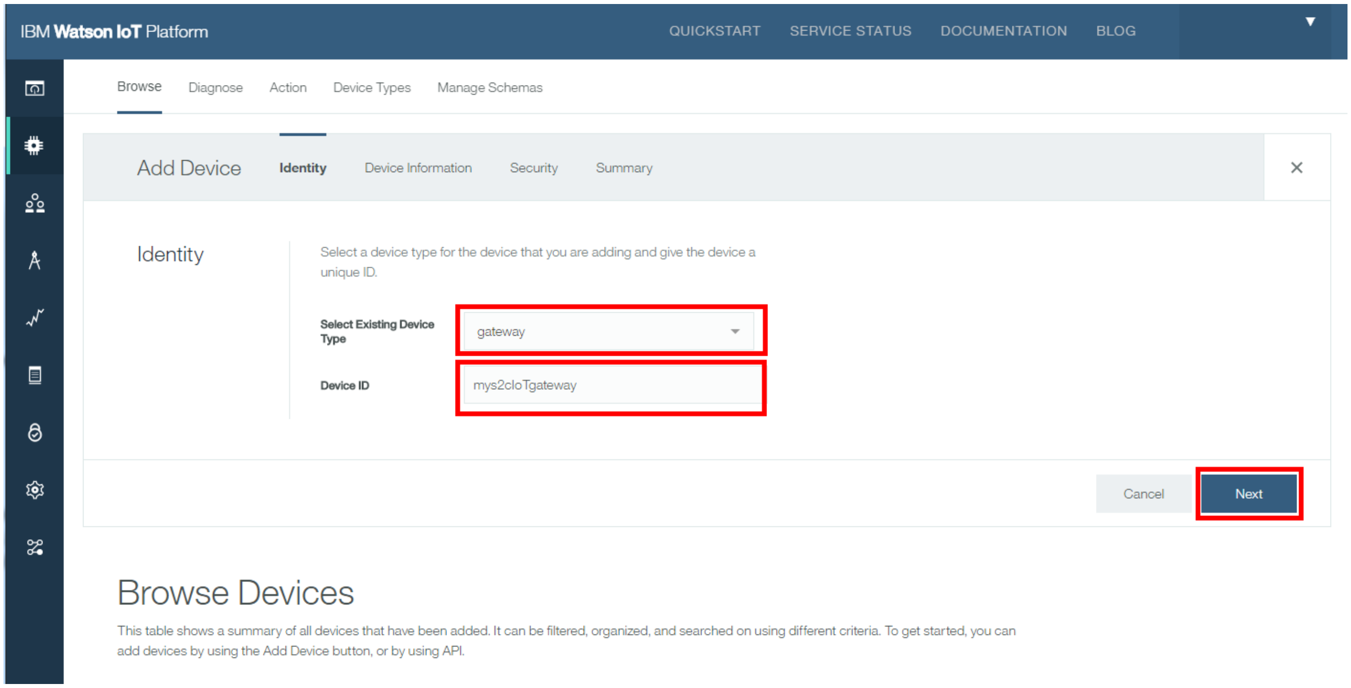
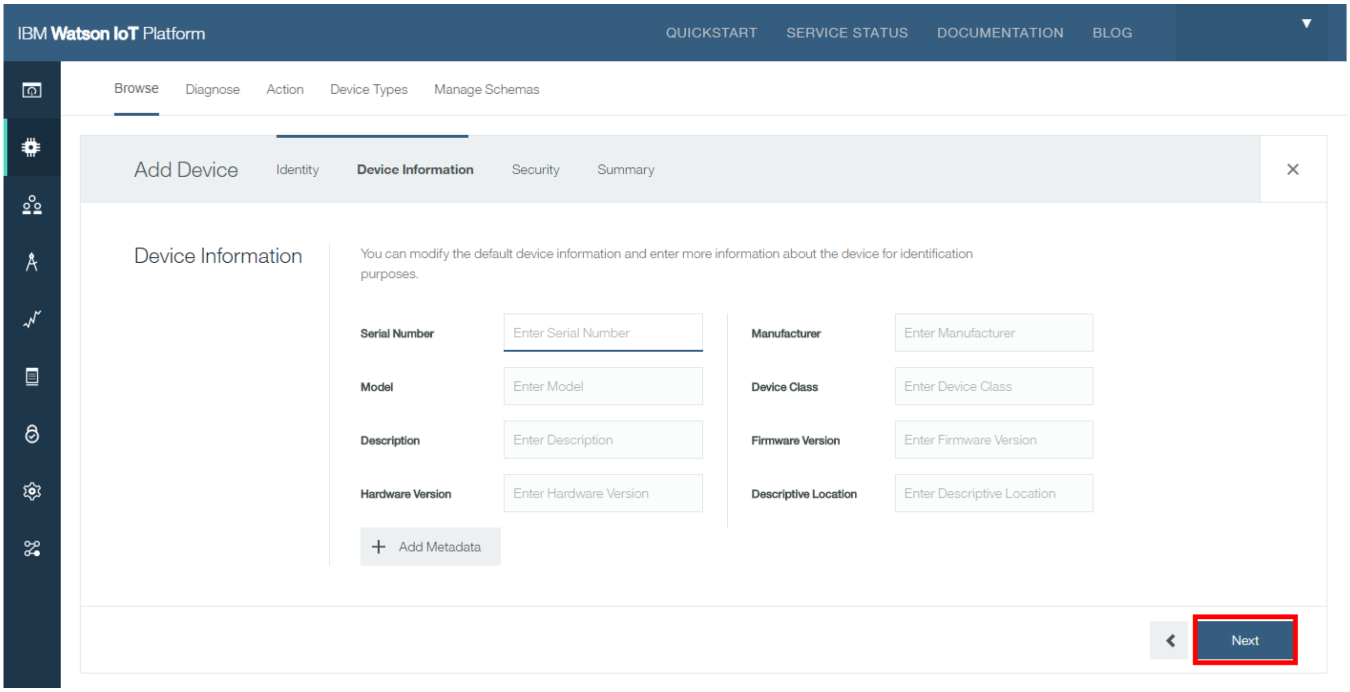

IBM Watson IoT Platform
QUICKSTART
SERVICE STATUS
DOCUMENTATION
BLOG
Browse
Diagnose
Action
Device Types
Manage Schemas
Add Device
Identity
Device Information
Security
Summary
×
Identity
Select a device type for the device that you are adding and give the device a unique ID.
Select Existing Device Type
gateway
Device ID
mys2cloTgateway
Cancel
Next
Browse Devices
This table shows a summary of all devices that have been added. It can be filtered, organized, and searched on using different criteria. To get started, you can add devices by using the Add Device button, or by using API.

Figure 42. Add Device Identity

7. Skip the Device Information and click next.



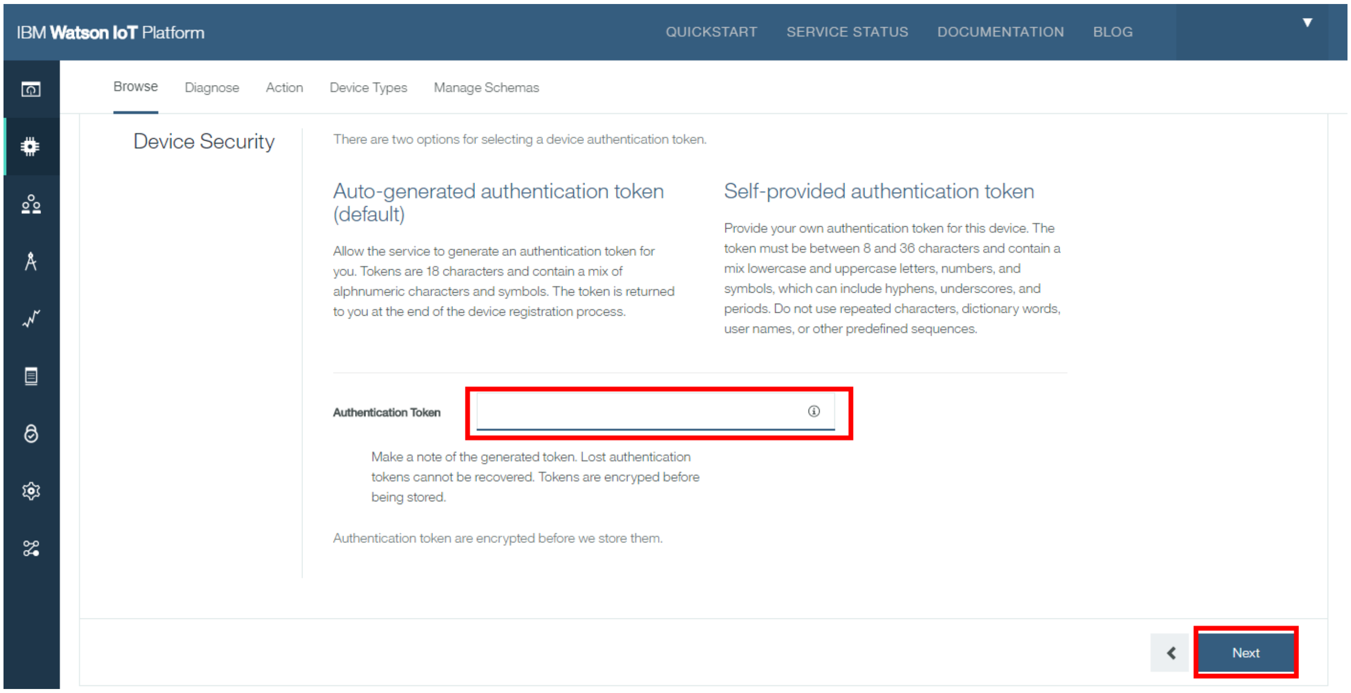
Device Information

You can modify the default device information and enter more information about the device for identification purposes.

Serial Number	Enter Serial Number	Manufacturer	Enter Manufacturer
Model	Enter Model	Device Class	Enter Device Class
Description	Enter Description	Firmware Version	Enter Firmware Version
Hardware Version	Enter Hardware Version	Descriptive Location	Enter Descriptive Location

Figure 43. Device ID

8. On the *Security* tab, fill out the *token* field. Make note of this token, as this will be used for authenticating the device to the cloud. Click *Next*.



Device Security

There are two options for selecting a device authentication token.

Auto-generated authentication token (default)

Allow the service to generate an authentication token for you. Tokens are 18 characters and contain a mix of alphnumeric characters and symbols. The token is returned to you at the end of the device registration process.

Self-provided authentication token

Provide your own authentication token for this device. The token must be between 8 and 36 characters and contain a mix lowercase and uppercase letters, numbers, and symbols, which can include hyphens, underscores, and periods. Do not use repeated characters, dictionary words, user names, or other predefined sequences.

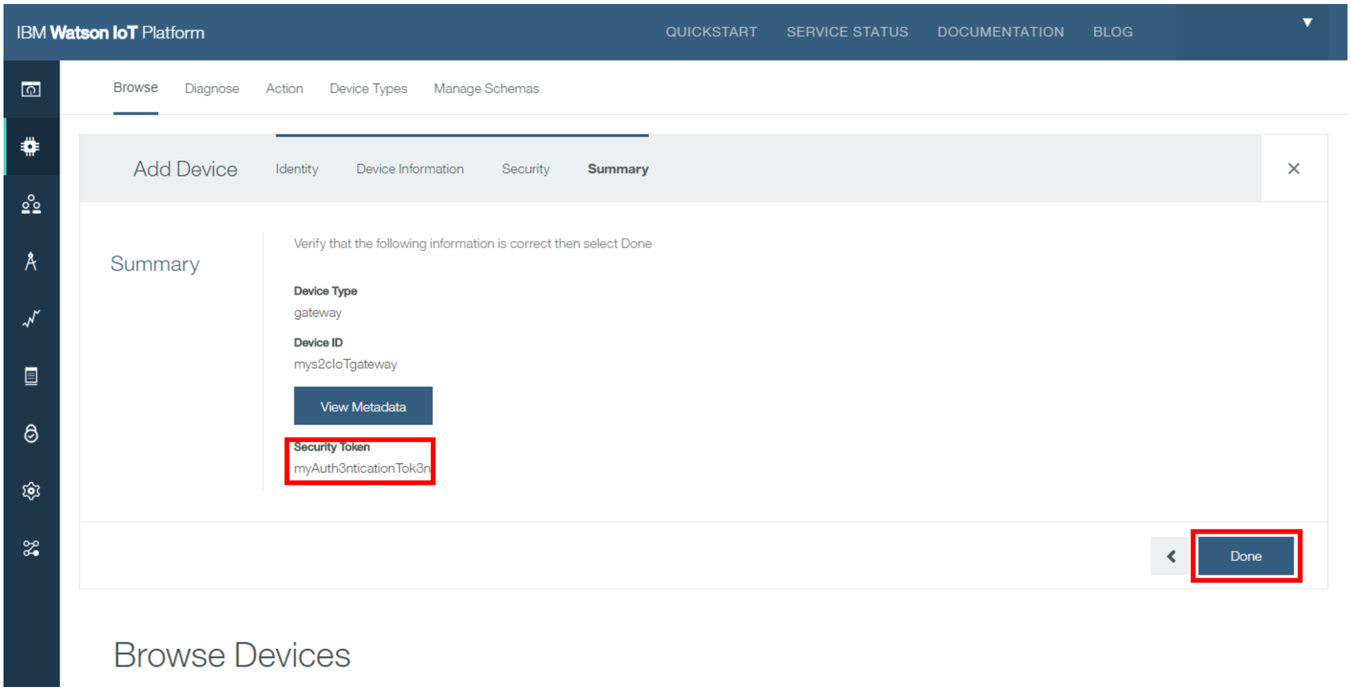
Authentication Token

Make a note of the generated token. Lost authentication tokens cannot be recovered. Tokens are encryped before being stored.

Authentication token are encrypted before we store them.

Figure 44. Authentication Token

9. Keep clicking *Next* until a summary of the device credentials and information shows. Take a screenshot of this page, as this will be the last time the *Authentication Token* is visible. Click on Done.



Summary

Verify that the following information is correct then select Done

Device Type
gateway

Device ID
mys2cloTgateway

Security Token
myAuth3nticationTok3n

Figure 45. Adding Device Summary

10. After adding the new device, go to security on the menu on the left as shown below.

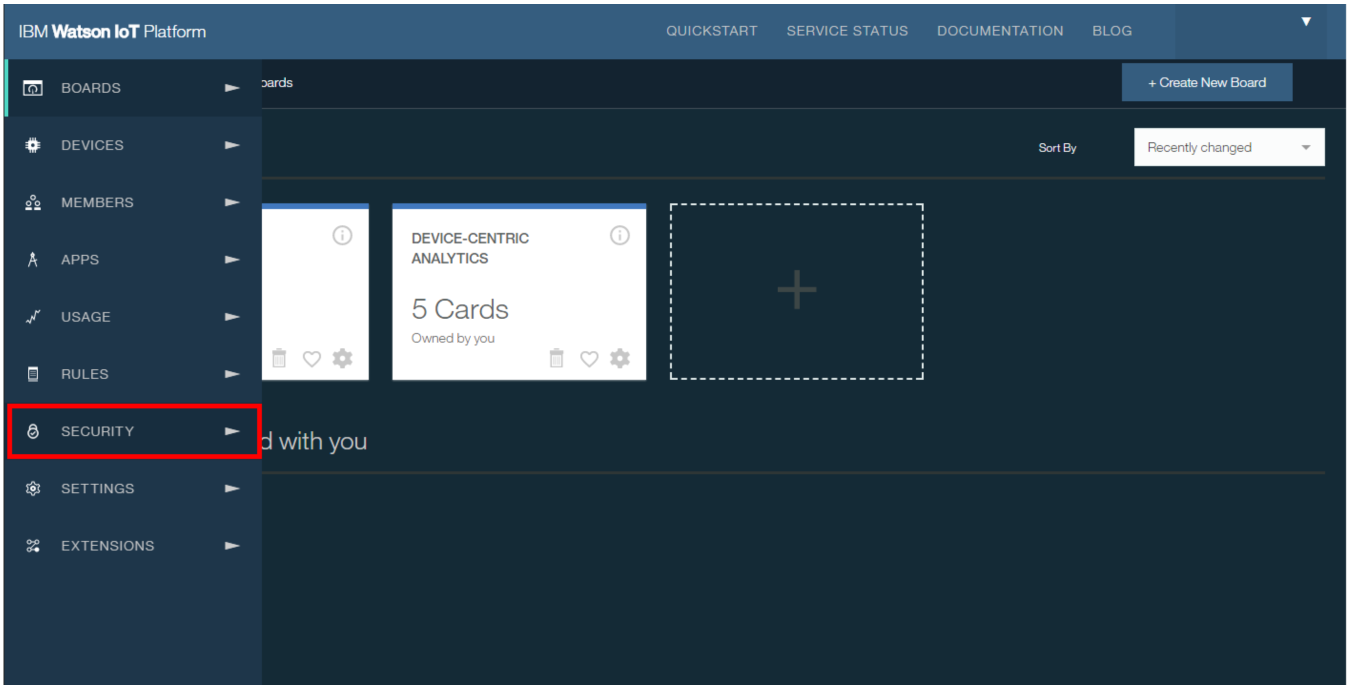
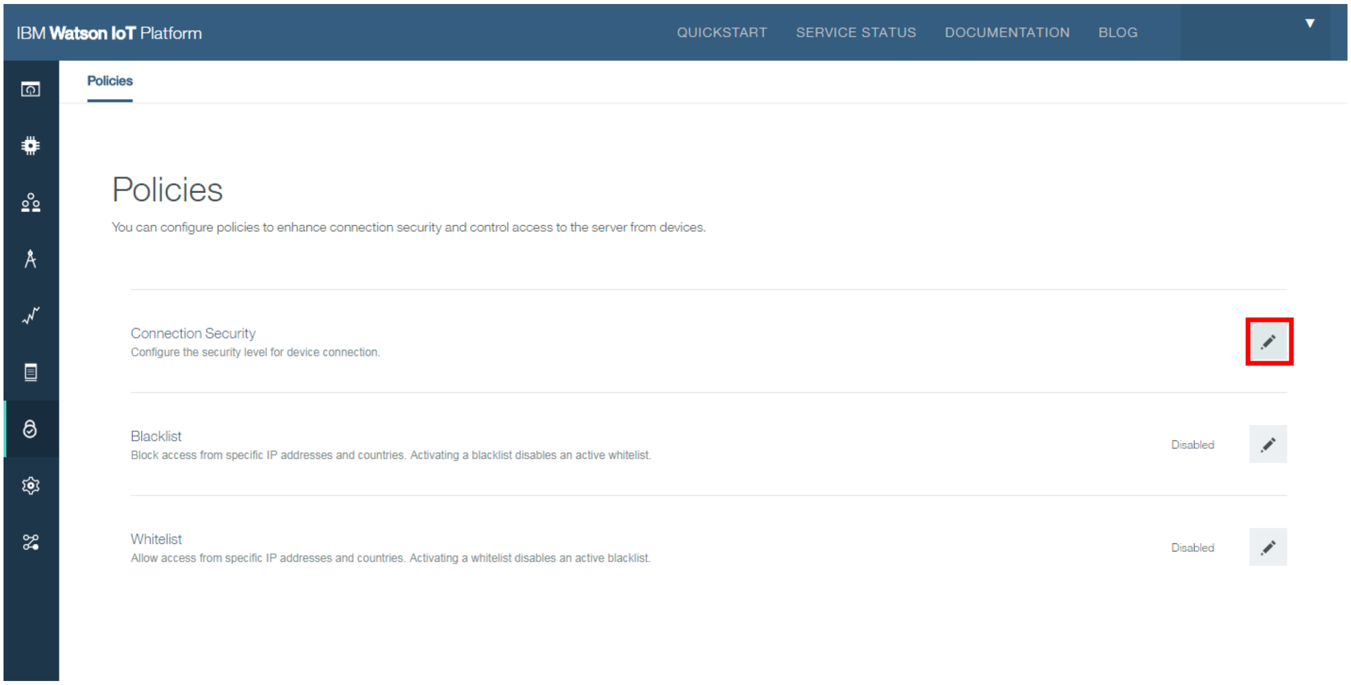


Figure 46. Security Menu

11. Click on the edit button for Connection Security.



Policies

You can configure policies to enhance connection security and control access to the server from devices.

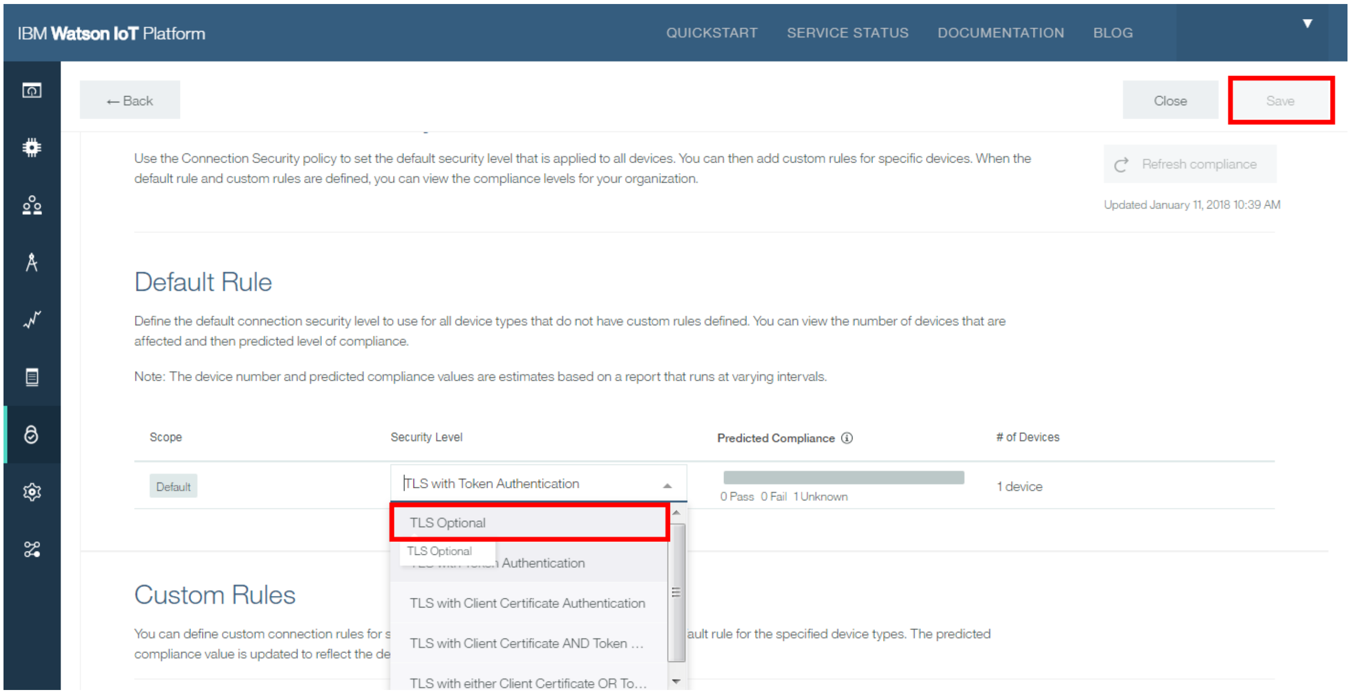
Connection Security
Configure the security level for device connection.

Blacklist
Block access from specific IP addresses and countries. Activating a blacklist disables an active whitelist.
Disabled

Whitelist
Allow access from specific IP addresses and countries. Activating a whitelist disables an active blacklist.
Disabled

Figure 47. Modifying Connection Security

12. Click on the drop down menu under security level and select TLS Optional, and then save the new configuration.



Use the Connection Security policy to set the default security level that is applied to all devices. You can then add custom rules for specific devices. When the default rule and custom rules are defined, you can view the compliance levels for your organization.

Updated January 11, 2018 10:39 AM

Default Rule

Define the default connection security level to use for all device types that do not have custom rules defined. You can view the number of devices that are affected and then predicted level of compliance.

Note: The device number and predicted compliance values are estimates based on a report that runs at varying intervals.

Scope	Security Level	Predicted Compliance	# of Devices
Default	TLS with Token Authentication	0 Pass 0 Fail 1 Unknown	1 device

TLS Optional
TLS Optional
TLS with Token Authentication
TLS with Client Certificate Authentication
TLS with Client Certificate AND Token ...
TLS with either Client Certificate OR To...

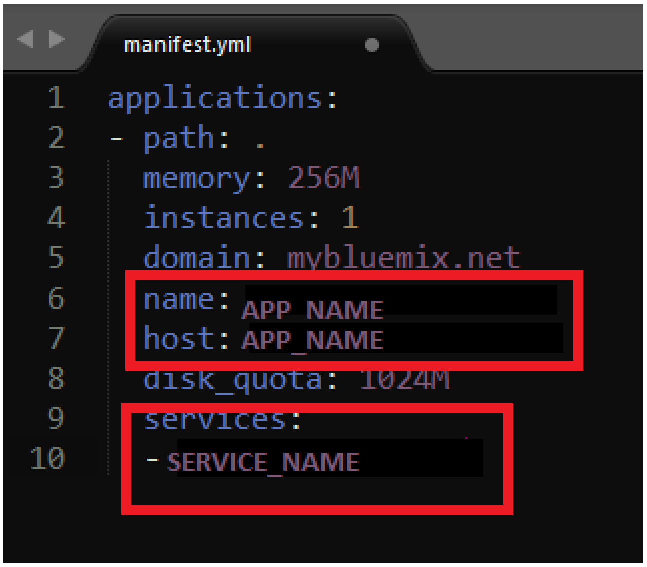
Custom Rules

You can define custom connection rules for s... ault rule for the specified device types. The predicted compliance value is updated to reflect the de...

Figure 48. Save New Security Rule

3.2.1.1.8 Set Up Node.js Cloud Foundry App

1. Locate *C:\<S2C Repo Directory>\tidc01002\examples* .
2. Open the *ibm_cloud_application* folder, and open the *Manifest.yml* with a text editor. Replace the name and *services* fields with the *name* of the cloud foundry app and Watson IoT platform.

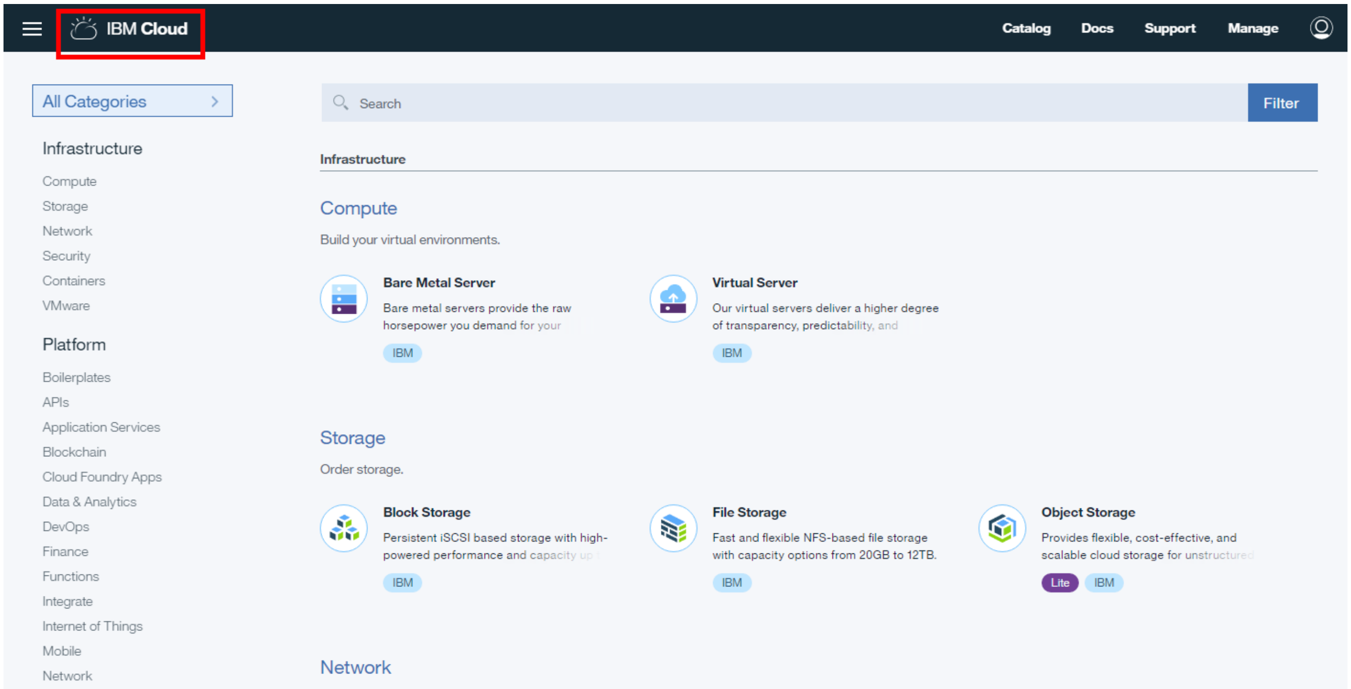


```
applications:
- path: .
  memory: 256M
  instances: 1
  domain: mybluemix.net
  name: APP_NAME [redacted]
  host: APP_NAME [redacted]
  disk_quota: 1024M
  services:
  - SERVICE_NAME [redacted]
```

Figure 49. Manifest

3. Open a command console and navigate to the IBM-Cloud-Dashboard folder (*cd C:\<S2C Repo Directory>\tidc01002\examples\ibm_cloud_application*).
4. Type in **cf api https://api.ng.bluemix.net**
5. Log in to the created account: **cf login**
6. Push the code to the IBM cloud foundry app: **cf push**

7. Go back to the IBM Cloud *Dashboard* by clicking *IBM Cloud* on the top left.



Infrastructure

Compute

Build your virtual environments.

Bare Metal Server — Bare metal servers provide the raw horsepower you demand for your — IBM

Virtual Server — Our virtual servers deliver a higher degree of transparency, predictability, and — IBM

Storage

Order storage.

Block Storage — Persistent iSCSI based storage with high-powered performance and capacity up — IBM

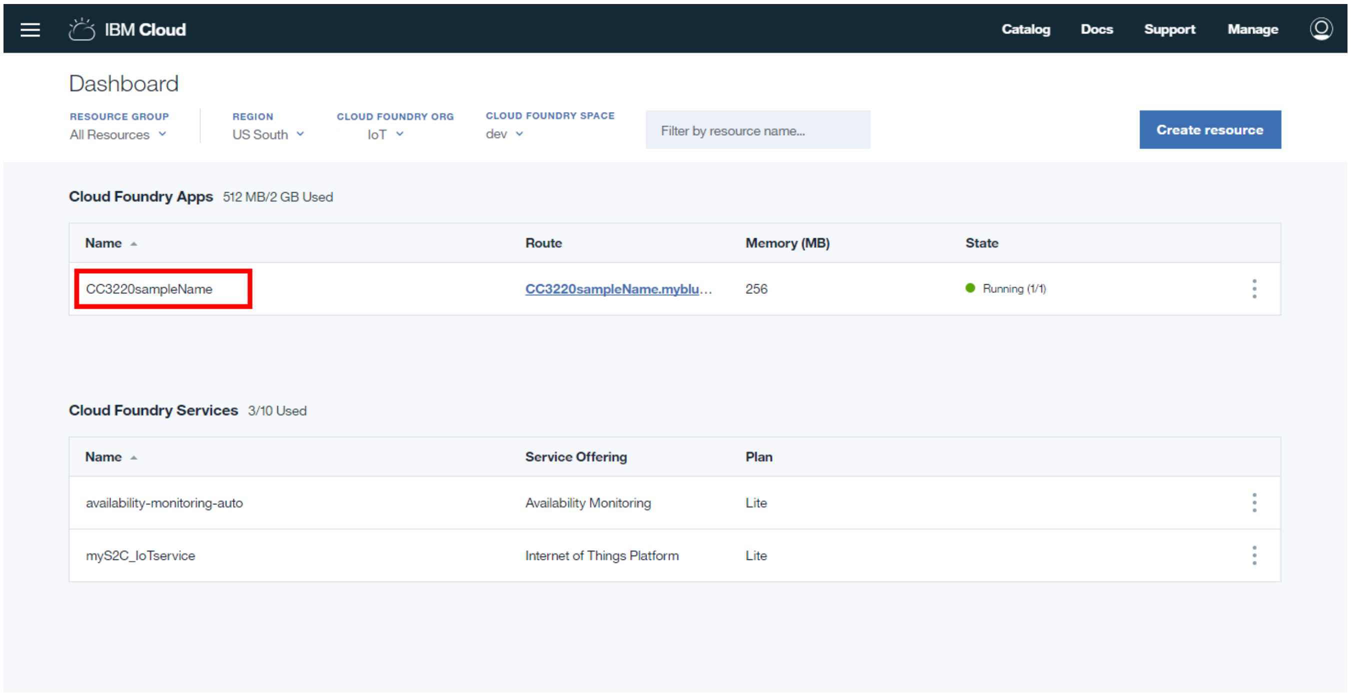
File Storage — Fast and flexible NFS-based file storage with capacity options from 20GB to 12TB. — IBM

Object Storage — Provides flexible, cost-effective, and scalable cloud storage for unstructured — Lite IBM

Network

Figure 50. Dashboard

8. The cloud foundry apps and IoT services created on the previous on the dashboard will be visible. Click on the cloud foundry app.



RESOURCE GROUP: All Resources
REGION: US South
CLOUD FOUNDRY ORG: IoT
CLOUD FOUNDRY SPACE: dev

Cloud Foundry Apps 512 MB/2 GB Used

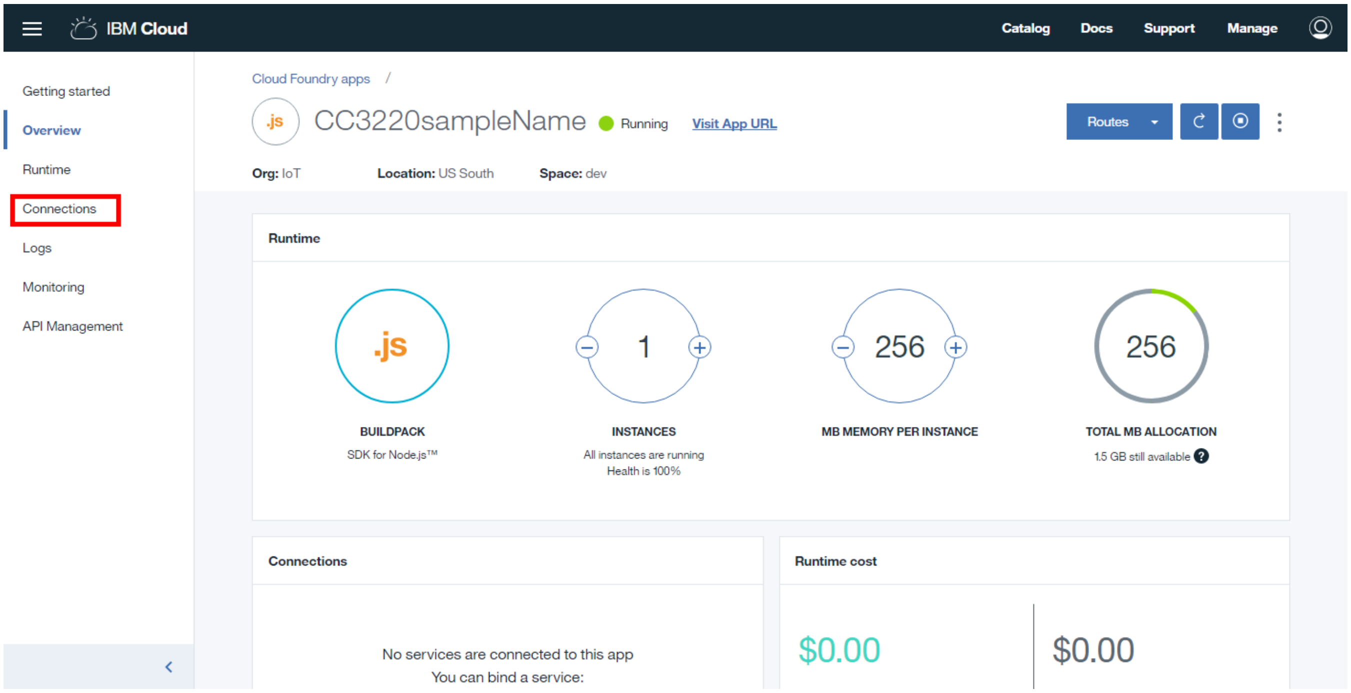
Name	Route	Memory (MB)	State
CC3220sampleName	CC3220sampleName.myblu...	256	● Running (1/1)

Cloud Foundry Services 3/10 Used

Name	Service Offering	Plan
availability-monitoring-auto	Availability Monitoring	Lite
myS2C_IoTservice	Internet of Things Platform	Lite

Figure 51. Cloud Foundry App

9. Click on the *Connections* tab to the left.

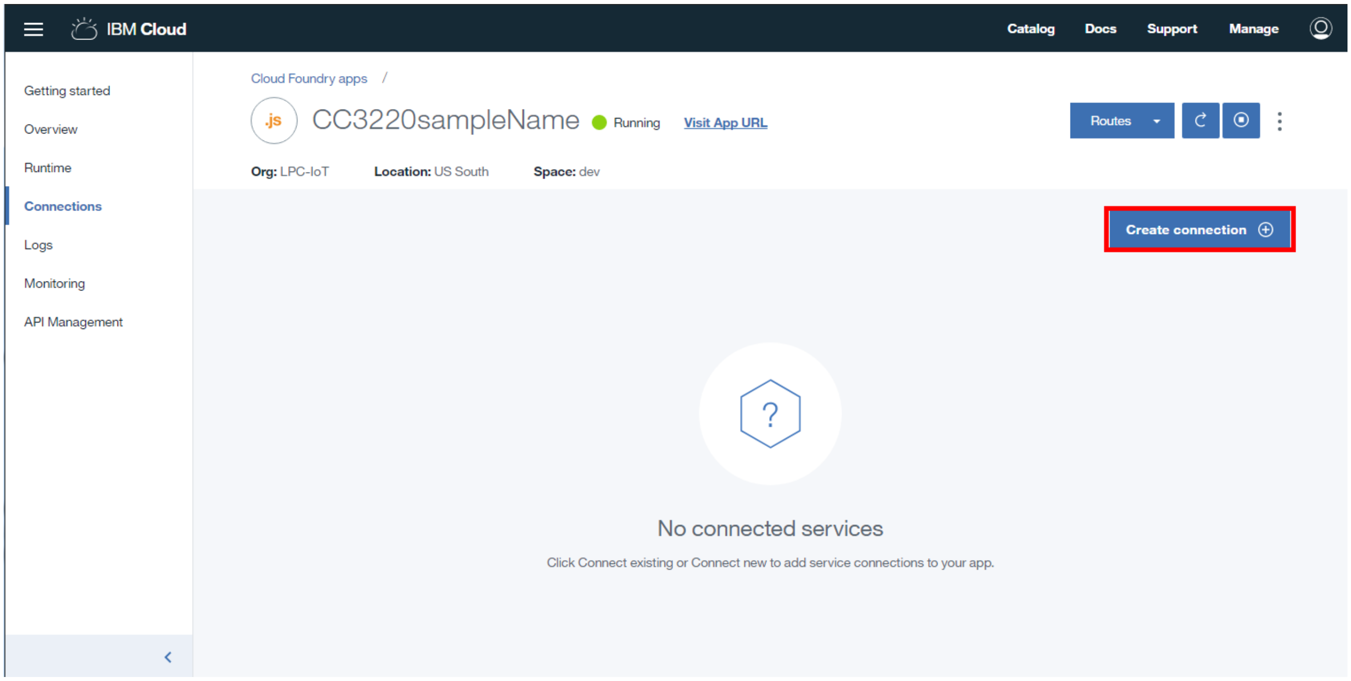


No services are connected to this app

You can bind a service:

Figure 52. Connections

10. Click on the *View credentials* button. This page shows the information required to establish a connection between the cloud front end and the back end server. Screenshot or save the information for later.

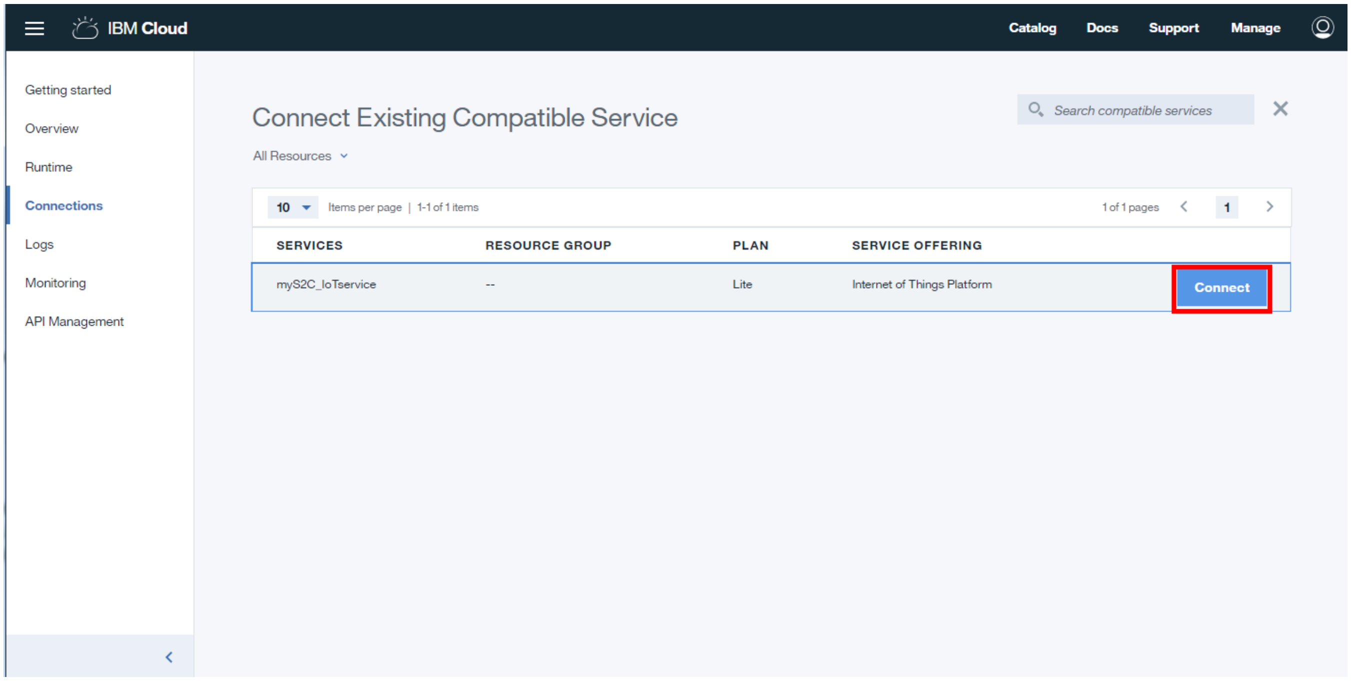


No connected services

Click Connect existing or Connect new to add service connections to your app.

Figure 53. View Credentials

11. Select the service previously created in this guide and click the connect button.

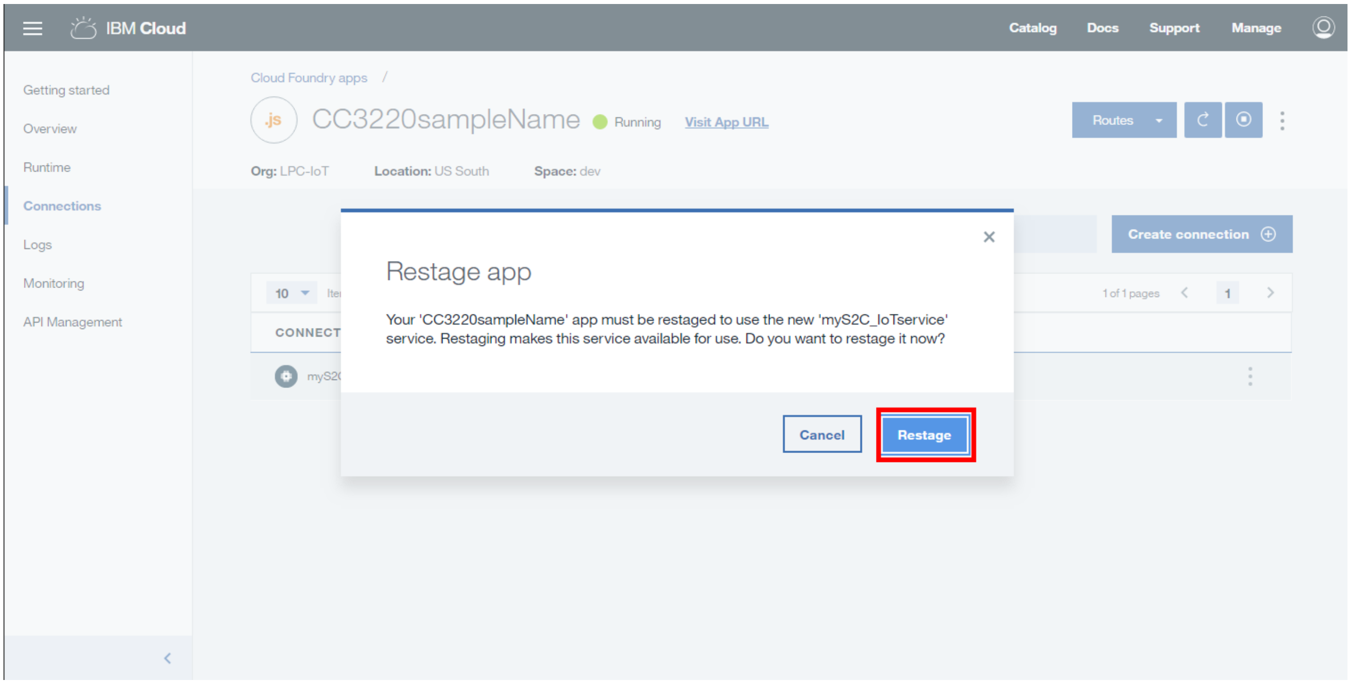


Connect Existing Compatible Service

SERVICES	RESOURCE GROUP	PLAN	SERVICE OFFERING
myS2C_IoTservice	--	Lite	Internet of Things Platform

Figure 54. Connect Service

12. Restage the app.



Restage app

Your 'CC3220sampleName' app must be restaged to use the new 'myS2C_IoTservice' service. Restaging makes this service available for use. Do you want to restage it now?

Figure 55. Restage the App

13. Click on the right side of the service and select View Credentials.

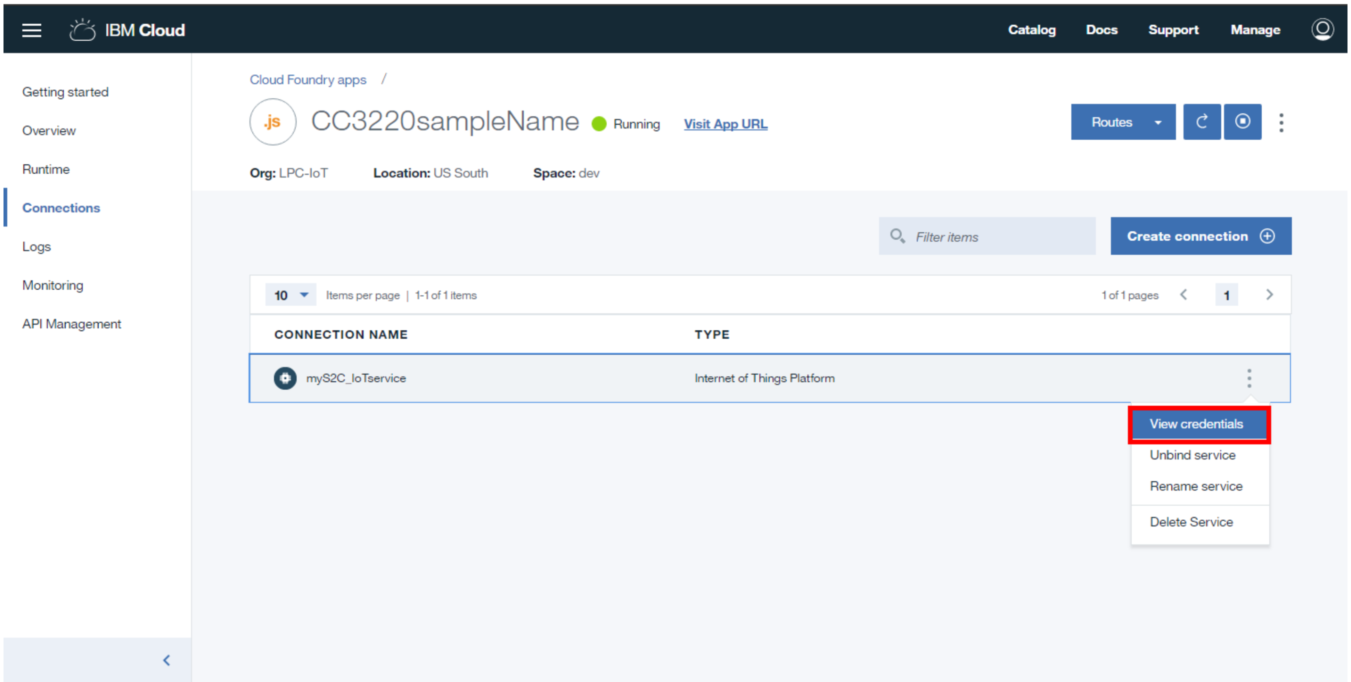
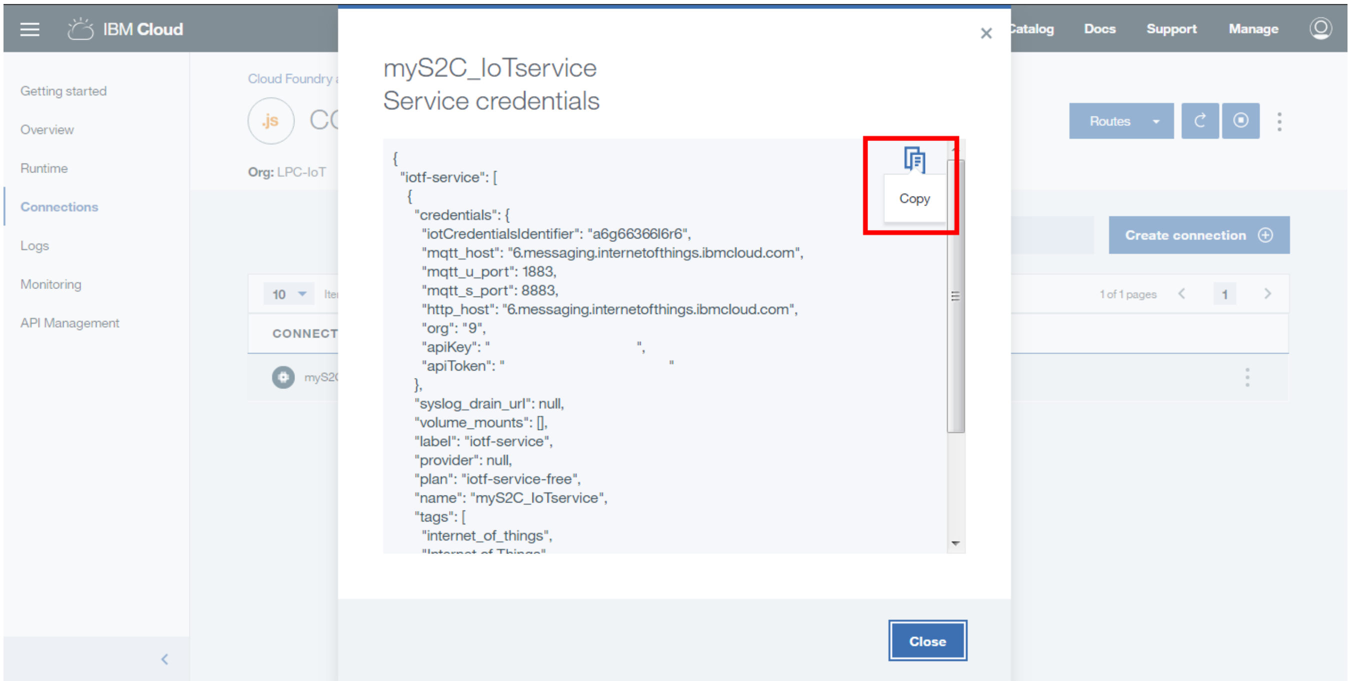


Figure 56. View Service Credentials

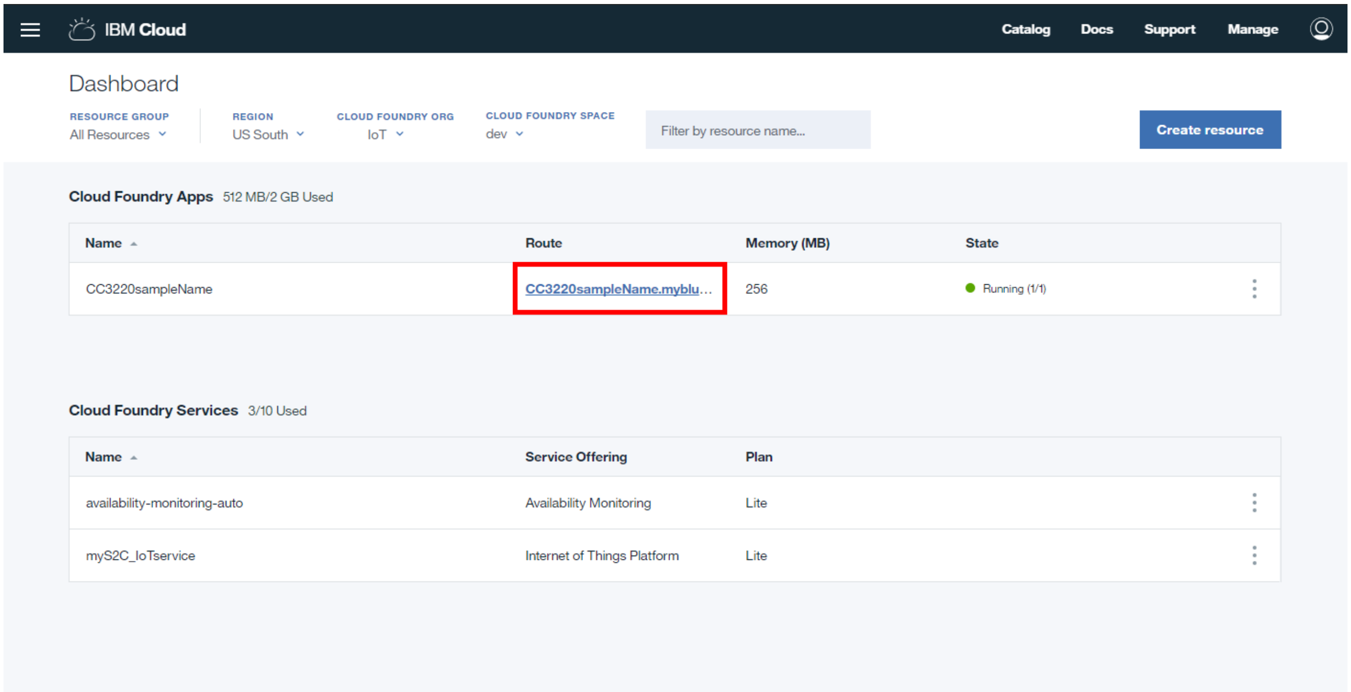
14. Take note of the Service credentials by clicking on the copy button and paste them into a notepad since they will be used in the next steps.



```
{
 "iotf-service": [
  {
   "credentials": {
    "iotCredentialsIdentifier": "a6g66366l6r6",
    "mqtt_host": "6.messaging.internetofthings.ibmcloud.com",
    "mqtt_u_port": 1883,
    "mqtt_s_port": 8883,
    "http_host": "6.messaging.internetofthings.ibmcloud.com",
    "org": "9",
    "apiKey": "                    ",
    "apiToken": "                    "
   },
   "syslog_drain_url": null,
   "volume_mounts": [],
   "label": "iotf-service",
   "provider": null,
   "plan": "iotf-service-free",
   "name": "myS2C_IoTservice",
   "tags": [
    "internet_of_things",
    "Internet of Things"
```

Figure 57. Copy Service credentials

15. Now go back to the Dashboard and open the webpagefor the Cloud Foundry App.The link can be found next to the cloud foundry app name.



RESOURCE GROUP: All Resources | REGION: US South | CLOUD FOUNDRY ORG: IoT | CLOUD FOUNDRY SPACE: dev

Cloud Foundry Apps 512 MB/2 GB Used

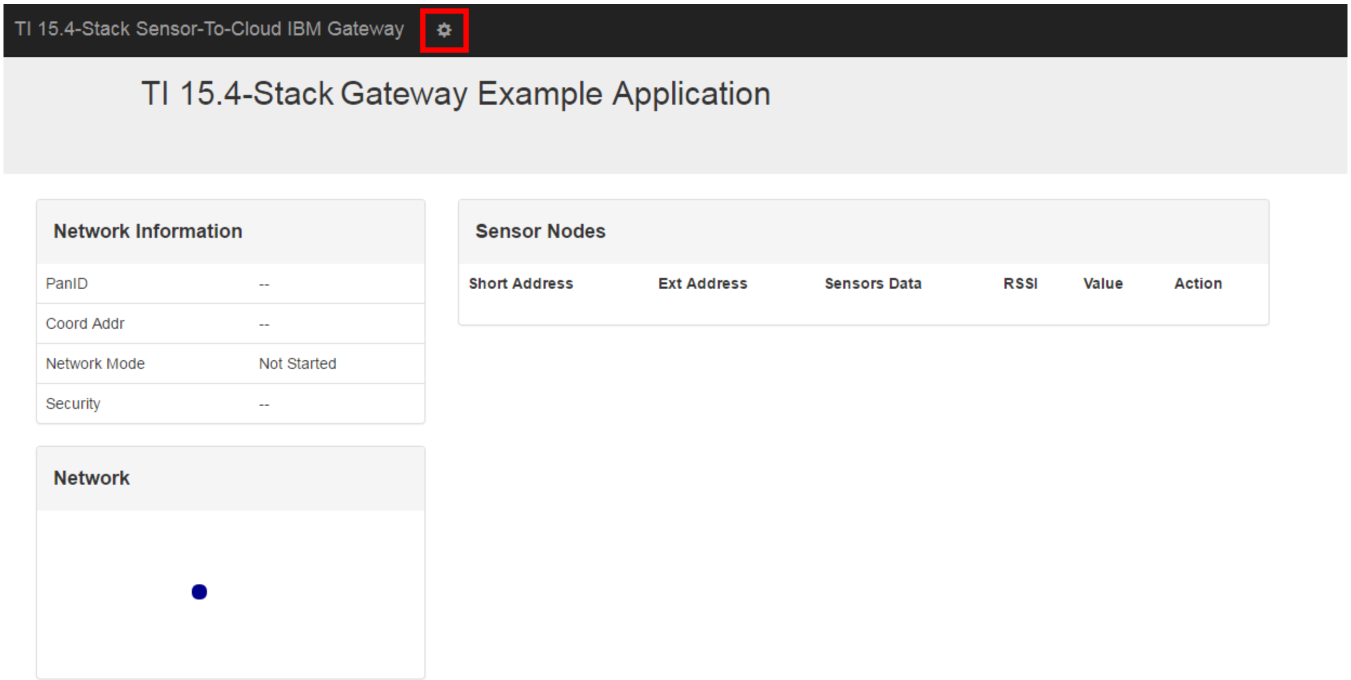
Name	Route	Memory (MB)	State
CC3220sampleName	CC3220sampleName.myblu...	256	● Running (1/1)

Cloud Foundry Services 3/10 Used

Name	Service Offering	Plan
availability-monitoring-auto	Availability Monitoring	Lite
myS2C_IoTservice	Internet of Things Platform	Lite

Figure 58. IBM® Web Page Link

16. If everything setup correctly, the dashboard will be visible. At this point, open the configuration menu located at the top.


TI 15.4-Stack Sensor-To-Cloud IBM Gateway
TI 15.4-Stack Gateway Example Application
Network

Network Information

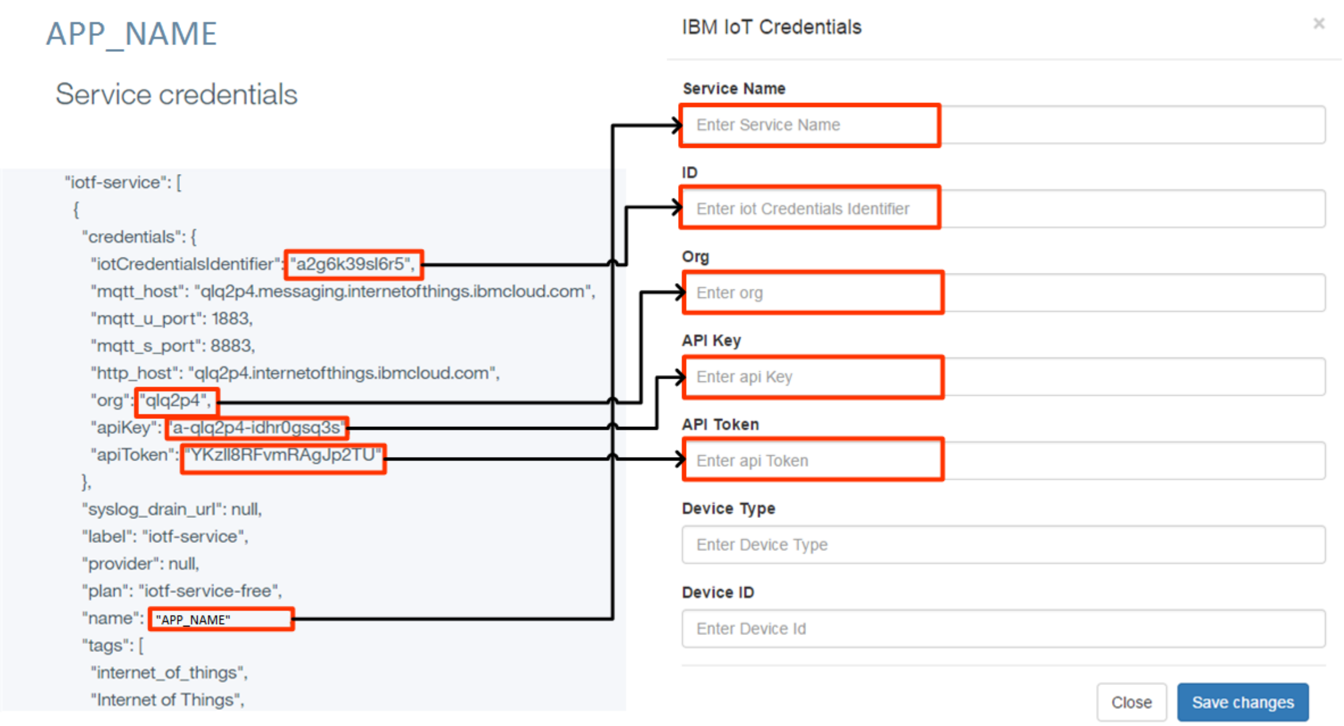
PanID	--
Coord Addr	--
Network Mode	Not Started
Security	--

Sensor Nodes

Short Address	Ext Address	Sensors Data	RSSI	Value	Action

Figure 59. Sensor2Cloud Front End

17. A form will pop up. Use the information saved in step 15 to fill out the form. For the *Device Type* and *Device ID*, use the information entered in Figure 60. Save the changes, and close when done.



```
"iotf-service": [
 {
  "credentials": {
   "iotCredentialsIdentifier": "a2g6k39sl6r5",
   "mqtt_host": "qlq2p4.messaging.internetofthings.ibmcloud.com",
   "mqtt_u_port": 1883,
   "mqtt_s_port": 8883,
   "http_host": "qlq2p4.internetofthings.ibmcloud.com",
   "org": "qlq2p4",
   "apiKey": "a-qlq2p4-idhr0gsq3s"
   "apiToken": "YKzll8RFvmRAgJp2TU"
  },
  "syslog_drain_url": null,
  "label": "iotf-service",
  "provider": null,
  "plan": "iotf-service-free",
  "name": "APP_NAME"
  "tags": [
   "internet_of_things",
   "Internet of Things",
```

Figure 60. IBM® IoT Credentials

3.2.1.1.9 Run and Use the Gateway

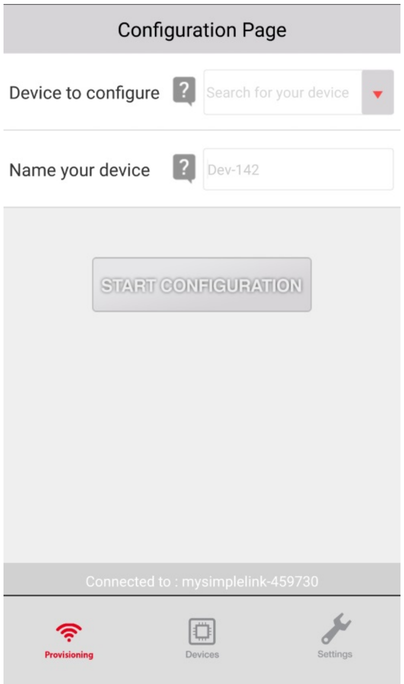
Before getting started with the instructions on how to run the gateway, make sure that the CC3220SF is plugged into the PC and that a serial console has been oppened on the serial port assigned to the CC3220SF LaunchPad.

There are two ways to get the S2C Gateway up an running. The first method is described in Section 3.2.1.1.9.1, which explains how to provision the CC3220SF LaunchPad to a WiFi Network from the Simple Link Starter Pro App. The second method is explained in Section 3.2.1.1.9.2; this method uses the built-in, local provisioning web page.

3.2.1.1.9.1 Using the SimpleLink™ Starter Pro App

This section assumes that either the IOS or the Android app is already installed on the user's mobile phone. If not, install the app now.

1. Launch the SimpleLink Starter Pro App from the phone.
2. If the device is not found automatically by the app, go to *Device to configure*, and tap on *search for your device*.



Configuration Page

Device to configure ? Search for your device

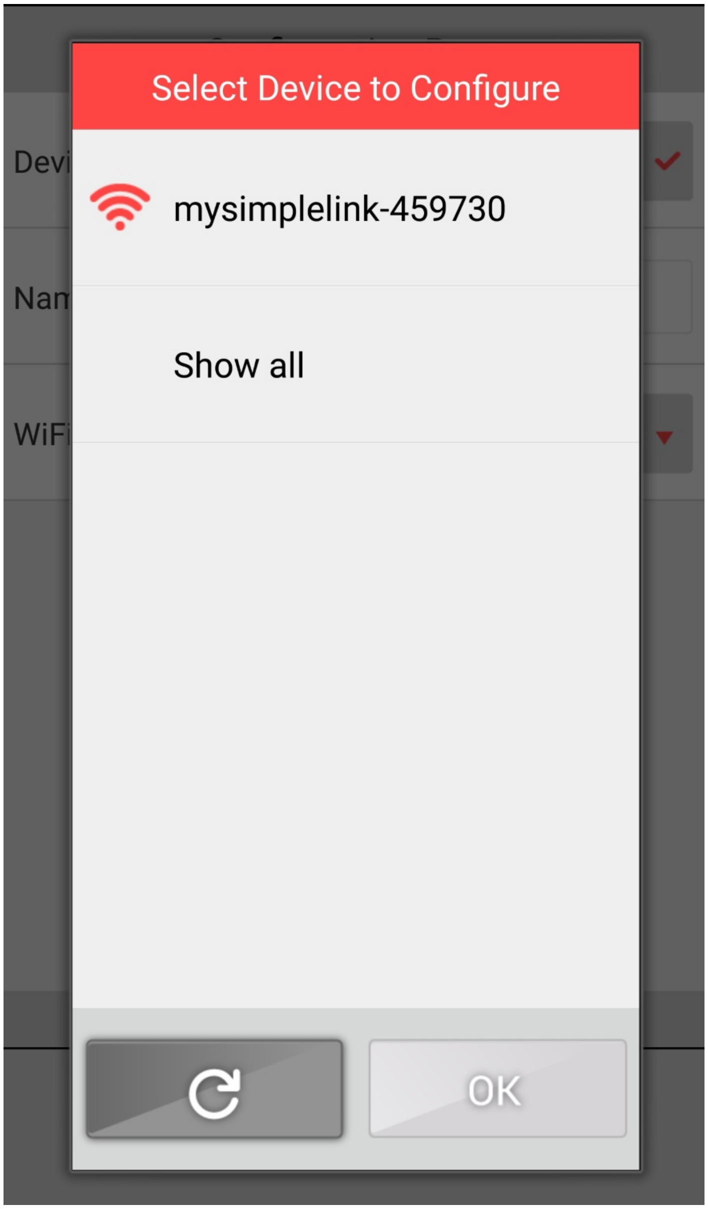
Name your device ? Dev-142

Connected to : mysimplelink-459730

Figure 61. Configuration Page

3. Wait for the app to find the device to connect. The name should be something like *mysimplelink-XXXX*.

4. Select the device to connect, and tap *OK*.



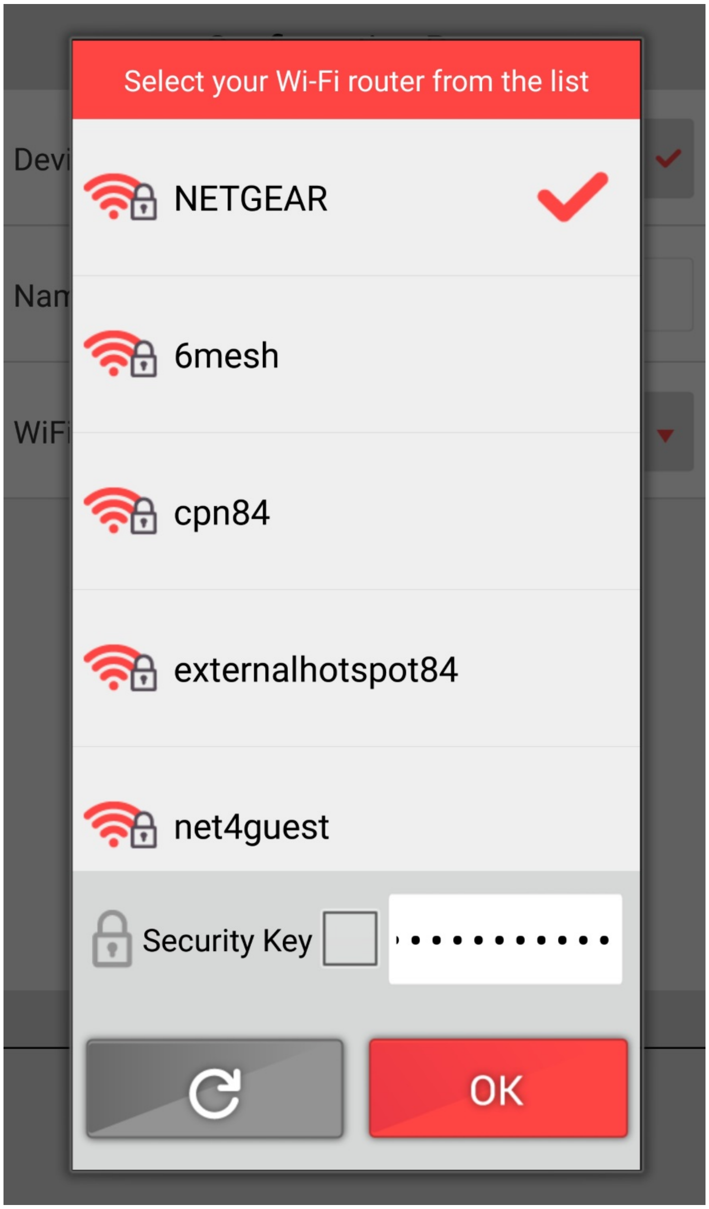
Select Device to Configure

mysimplelink-459730

Show all

Figure 62. Select Device to Configure

5. Select the desired Wi-Fi network to connect with the CC3220SF LaunchPad. Enter the password, tap *OK*, and then tap *Start Configuration*.



Select your Wi-Fi router from the list

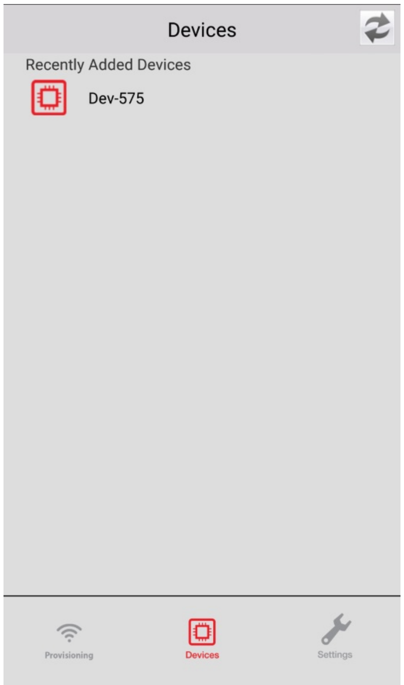
- NETGEAR ✓
- 6mesh
- cpn84
- externalhotspot84
- net4guest

Security Key □ ••••••••••••

Figure 63. Select Wi-Fi Router

6. After the configuration is done, make sure the phone is connected to the same Wi-Fi network that the CC3220SF is connected.

7. Once connected, select the device from the device list.



Devices

Recently Added Devices

Dev-575

Figure 64. Devices

8. This will open a web page hosted by the device. If the device has already been provisioned to a Wi-Fi network then click on Sensor Dashboard, otherwise click on Configure WiFi Network and enter the WiFi access point credentials.

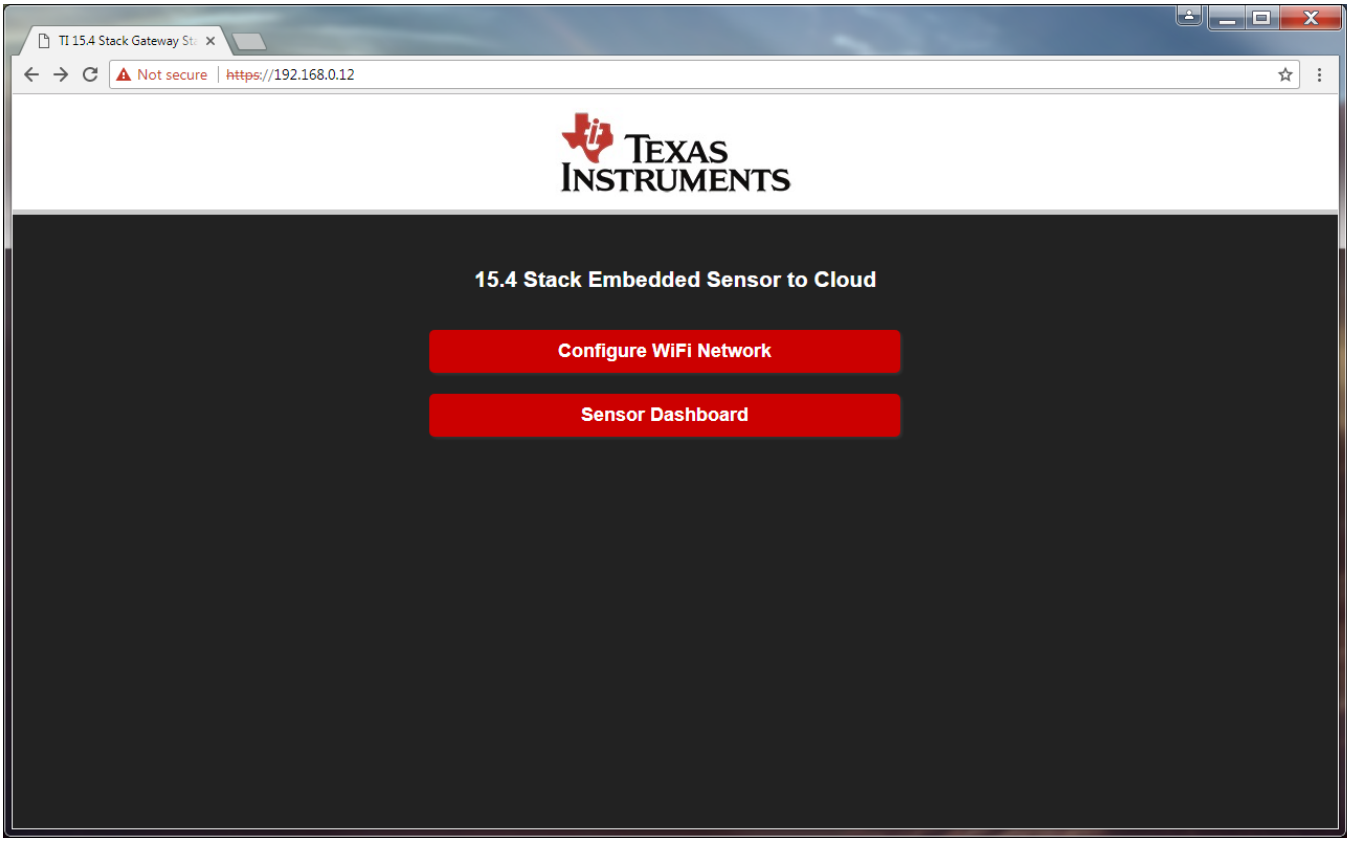
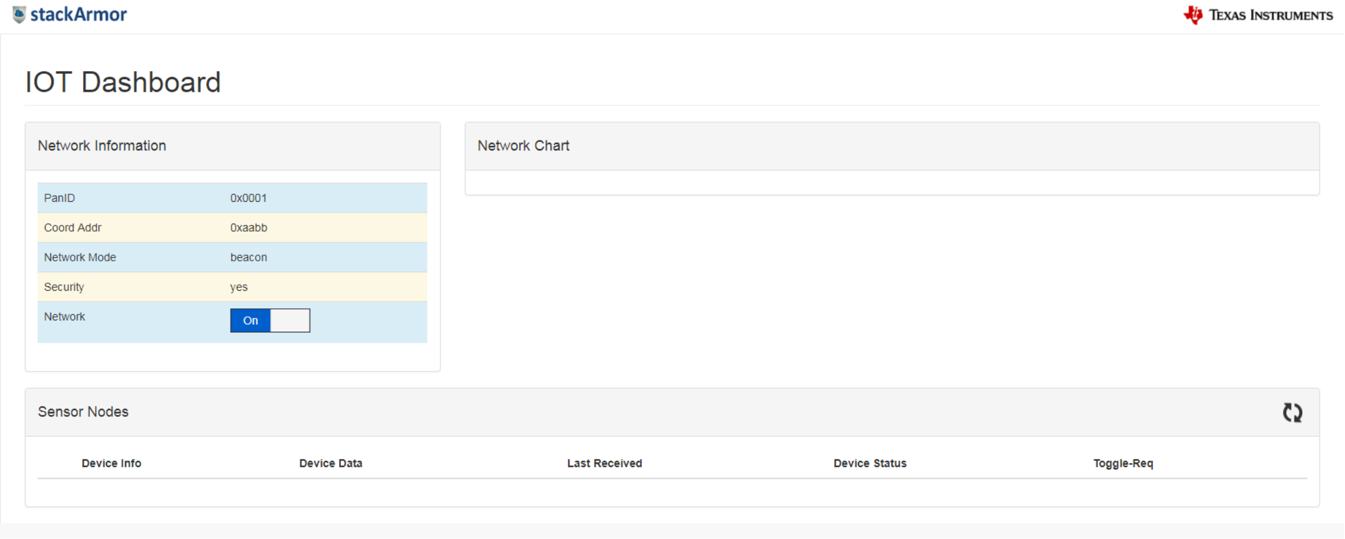


Figure 65. Local Web Server Start Page

NOTE: If the browser gives a warning about security certificates, ignore it, and continue to the web page.

9. *(AWS IoT Example Only)* If running the AWS Example and the gateway is connected to the Internet then got to the front end website provided by StackArmor, it should look something like this *http://iotdash.stackbuilder.us/#/dashboard/home?net=AWS_IOT_MY_THING_NAME* where *AWS_IOT_MY_THING_NAME* should match the thing name in aws_iot_config.h

   NOTE: if running the IBM Example then continue with the steps below.



IOT Dashboard

Network Information

PanID	0x0001
Coord Addr	0xaabb
Network Mode	beacon
Security	yes
Network	On

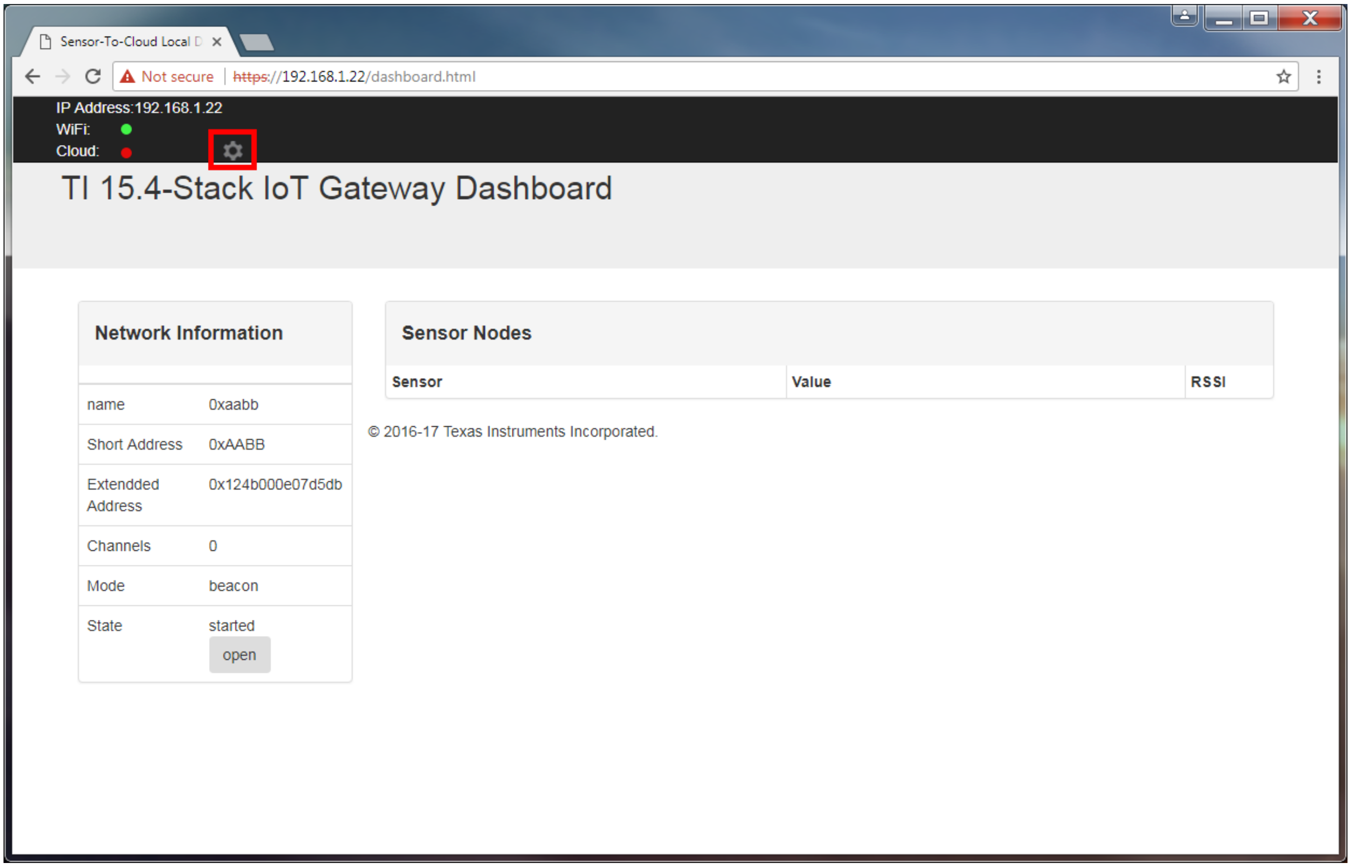
Network Chart

Sensor Nodes

Device Info	Device Data	Last Received	Device Status	Toggle-Req

Figure 66. AWS IoT Dashboard

10. *(IBM Example Only)*On the start page click on the Sensor Dashboard button. Once on the dashboard click on the settings button to enter the IBM Cloud configuration information.



IP Address:192.168.1.22
WiFi: ●
Cloud: ●

TI 15.4-Stack IoT Gateway Dashboard

Network Information

name	0xaabb
Short Address	0xAABB
Extendded Address	0x124b000e07d5db
Channels	0
Mode	beacon
State	started open

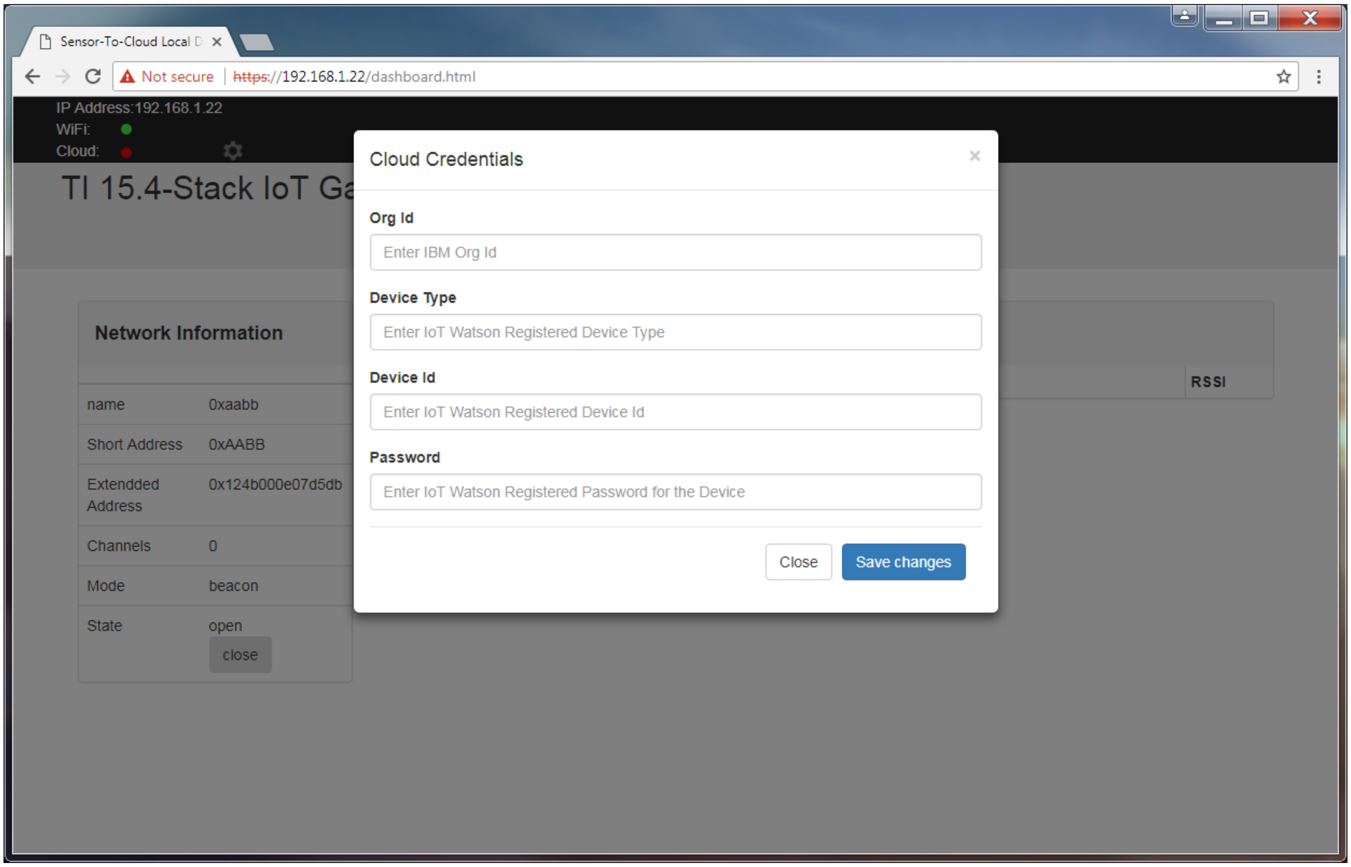
Sensor Nodes

Sensor	Value	RSSI

© 2016-17 Texas Instruments Incorporated.

Figure 67. Local Dashboard

11. *(IBM Example Only)* A form will pop up, fill out the form with the IBM Cloud account information. When complete, click *Save Changes* and then *Close*. If the Gateway was successful on connecting to the IBM Cloud then the Red dot next to "Cloud" on the dashboard should turn green.



Sensor-To-Cloud Local D ×

Not secure | https://192.168.1.22/dashboard.html

IP Address:192.168.1.22
WiFi: ●
Cloud: ●

TI 15.4-Stack IoT Ga

Network Information

name	0xaabb
Short Address	0xAABB
Extendded Address	0x124b000e07d5db
Channels	0
Mode	beacon
State	open close

RSSI

Cloud Credentials ×

Org Id
Enter IBM Org Id

Device Type
Enter IoT Watson Registered Device Type

Device Id
Enter IoT Watson Registered Device Id

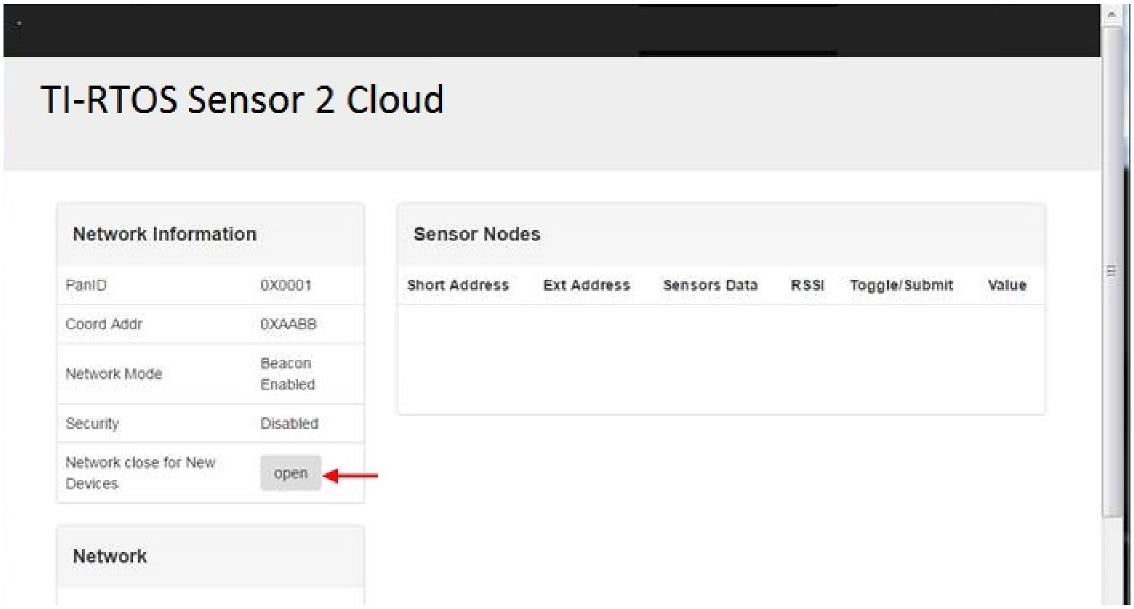
Password
Enter IoT Watson Registered Password for the Device

Close | Save changes

Figure 68. Cloud Credentials Form

12. *(IBM Example Only)* Go to the IBM Cloud Foundry App URL provided on the IBM Cloud account, which looks something like *APP_NAME.mybluemix.net*.

13. Click the *open* button on the dashboard to allow sensors to join the network.



TI-RTOS Sensor 2 Cloud

Network Information

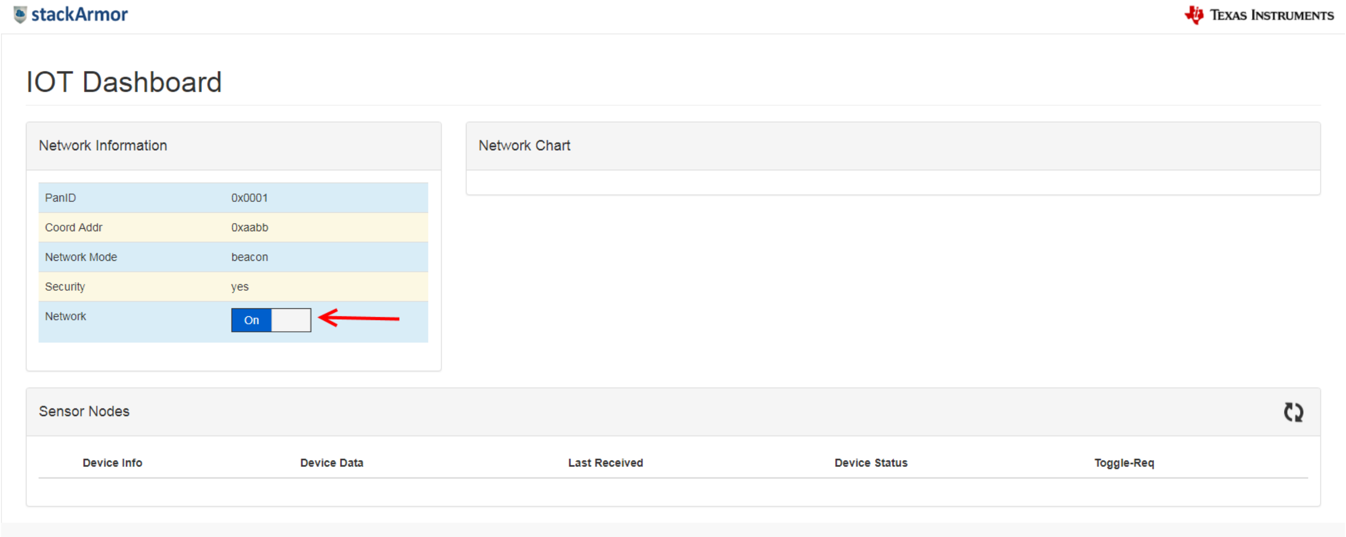
PanID	0X0001
Coord Addr	0XAABB
Network Mode	Beacon Enabled
Security	Disabled
Network close for New Devices	open

Network

Sensor Nodes

Short Address	Ext Address	Sensors Data	RSSI	Toggle/Submit	Value

Figure 69. IBM Cloud Dashboard Open Button



IOT Dashboard

Network Information

PanID	0x0001
Coord Addr	0xaabb
Network Mode	beacon
Security	yes
Network	On

Network Chart

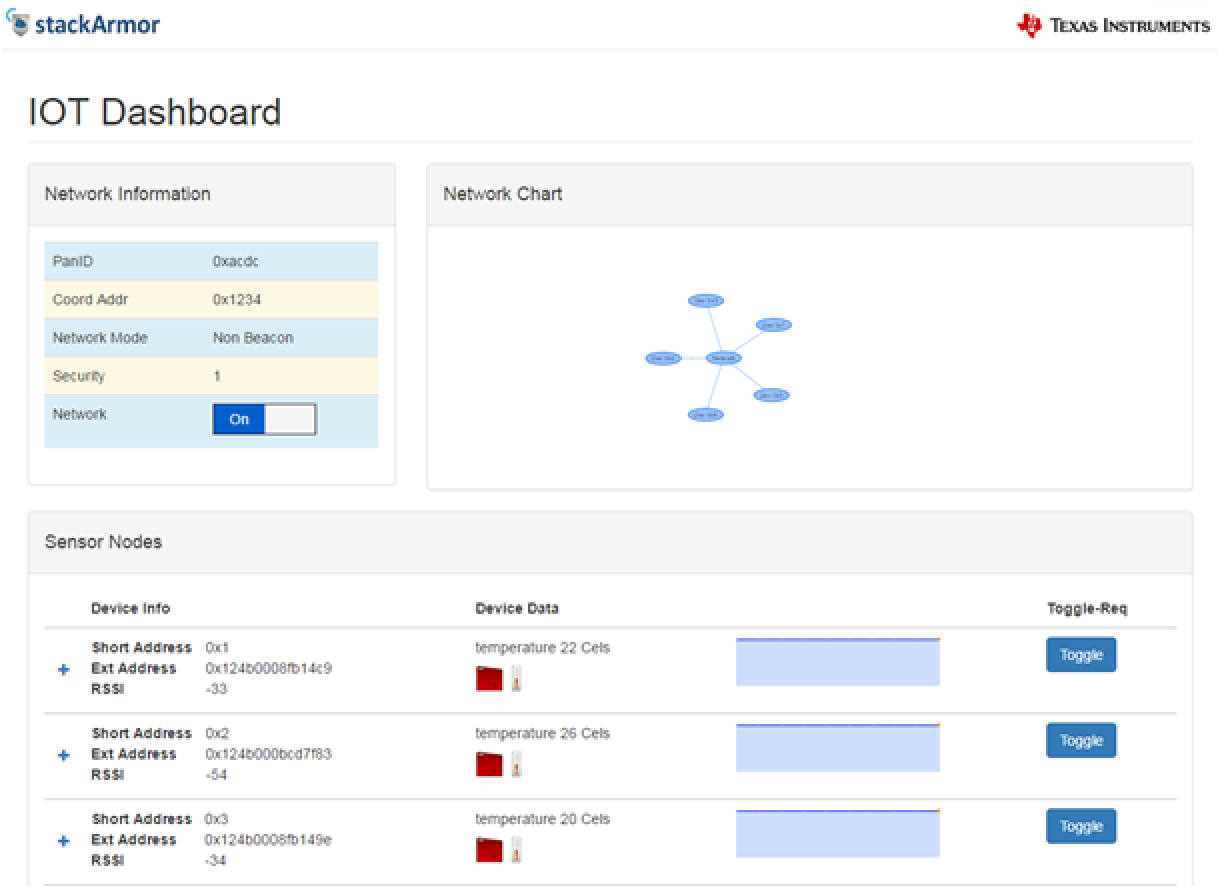
Sensor Nodes

Device Info	Device Data	Last Received	Device Status	Toggle-Req

Figure 70. AWS Cloud Dashboard Open Button

14. Apply power to the LaunchPad labeled *Sensor.*
15. Now the sensor should automatically start looking for a network. If paired with the network successfully, the *Sensor* board can be viewed and controlled from the web browser.

NOTE: If the device is not visible in the web browser, the device is most likely connected to another network. To solve this error, complete a factory reset on the sensor by pressing the reset button while holding the right button (BTN-2), and try again.



IOT Dashboard

Network Information

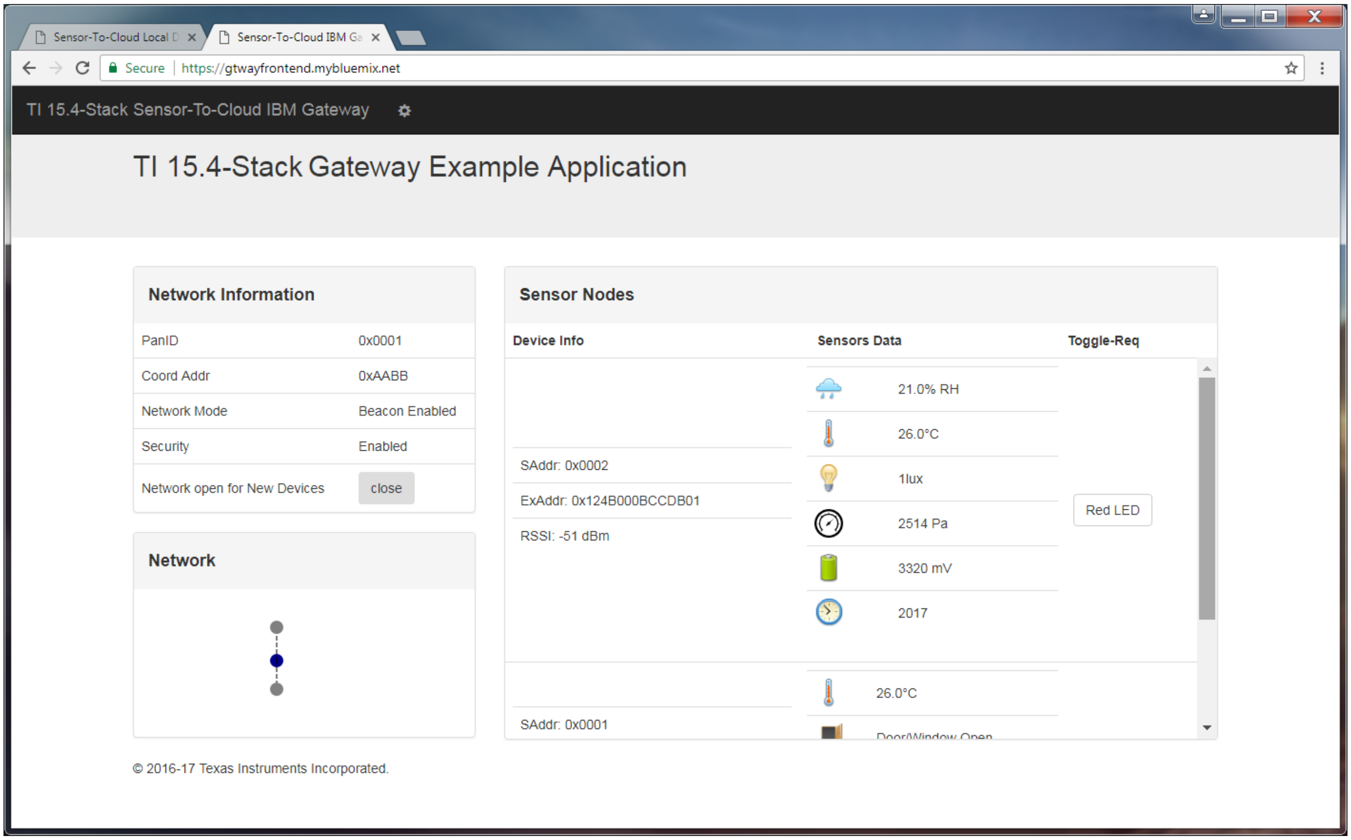
PanID	0xacdc
Coord Addr	0x1234
Network Mode	Non Beacon
Security	1
Network	On

Network Chart

Sensor Nodes

	Device Info		Device Data	Toggle-Req
+	Short Address Ext Address RSSI	0x1 0x124b0008fb14c9 -33	temperature 22 Cels	Toggle
+	Short Address Ext Address RSSI	0x2 0x124b000bcd7f83 -54	temperature 26 Cels	Toggle
+	Short Address Ext Address RSSI	0x3 0x124b0008fb149e -34	temperature 20 Cels	Toggle

Figure 71. AWS Front End After Devices Joined



Sensor-To-Cloud Local D ×
Sensor-To-Cloud IBM Ga ×
Secure | https://gtwayfrontend.mybluemix.net

TI 15.4-Stack Sensor-To-Cloud IBM Gateway

TI 15.4-Stack Gateway Example Application

Network Information

PanID	0x0001
Coord Addr	0xAABB
Network Mode	Beacon Enabled
Security	Enabled
Network open for New Devices	close

Network

Sensor Nodes

Device Info	Sensors Data	Toggle-Req
SAddr: 0x0002 ExAddr: 0x124B000BCCDB01 RSSI: -51 dBm	21.0% RH 26.0°C 1lux 2514 Pa 3320 mV 2017	Red LED
SAddr: 0x0001	26.0°C Door/Window Open	

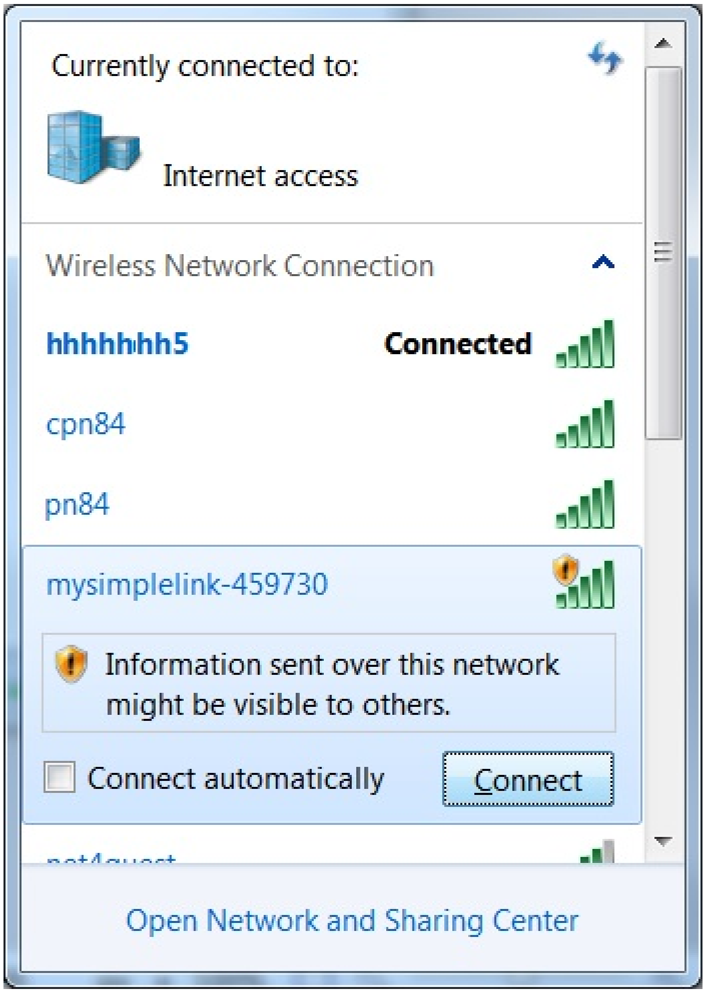
© 2016-17 Texas Instruments Incorporated.

Figure 72. Sensor2Cloud Front End

3.2.1.1.9.2 Using the Local Provisioning Web Page

NOTE: This is an alternate option if using the SimpleLink App is not desired.

1. Make sure the CC3220SF LaunchPad is powered on and that a serial console, like Tera Term, is opened to see the console output of the device.
2. On the PC search for Wi-Fi networks, and connect to the one broadcasted by the LaunchPad, which should look something like *mysimplelink-XXXX*.



Information sent over this network might be visible to others.

Figure 73. Connect to LaunchPad™

3. Open a browser window, and navigate to *mysimplelink.net* this should open the start page shown below.

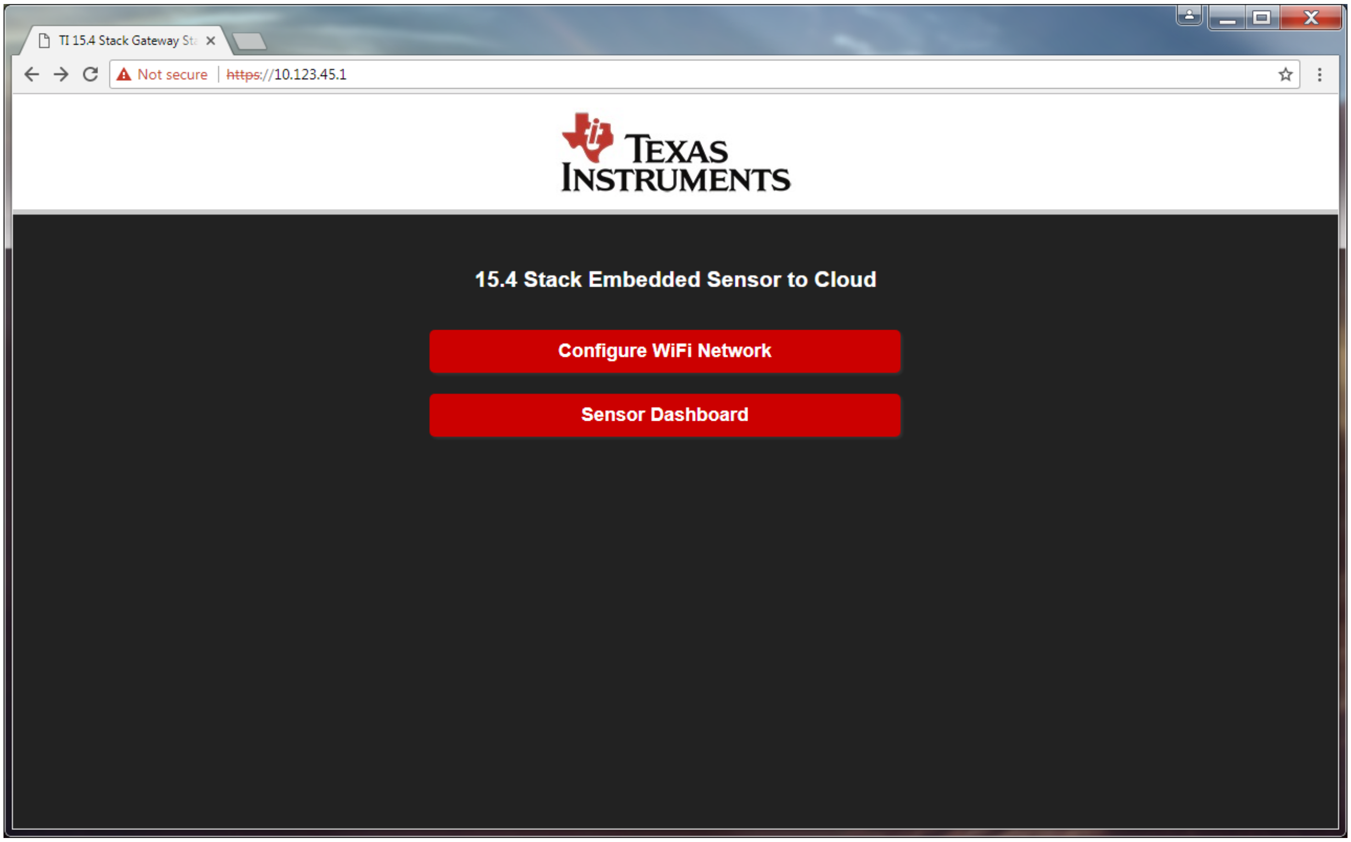
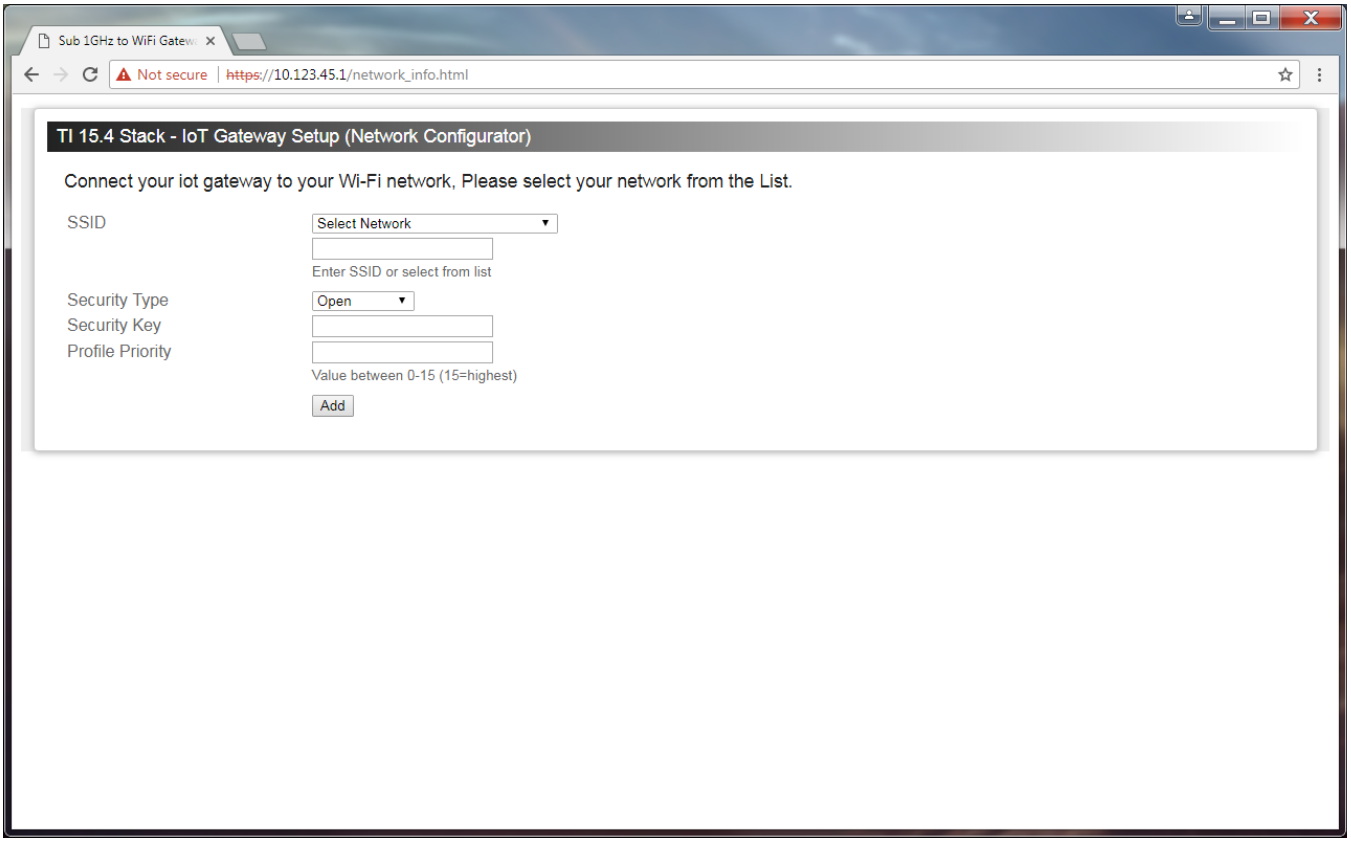


Figure 74. mysimplelink.net Local Home Page

4. Press the *Configure WiFi Network* button to go to the *Network Configuration* page.

   Note: Before entering the network information make sure to have a serial terminal open for the CC3220 gateway so you can record the IP address given to the gateway.



TI 15.4 Stack - IoT Gateway Setup (Network Configurator)

Connect your iot gateway to your Wi-Fi network, Please select your network from the List.

SSID: Select Network ▾

Enter SSID or select from list

Security Type: Open ▾

Security Key

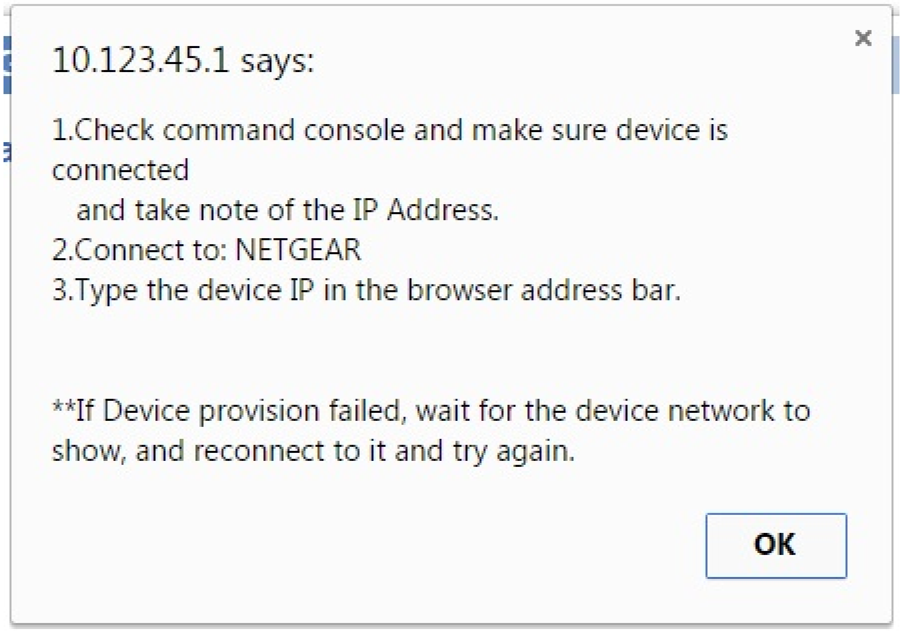
Profile Priority

Value between 0-15 (15=highest)

Add

Figure 75. Network Configuration Page

5. Enter the required information for the desired Wi-Fi access point, and click on the *Add* button. A pop-up window with instructions like Figure 76 will appear.



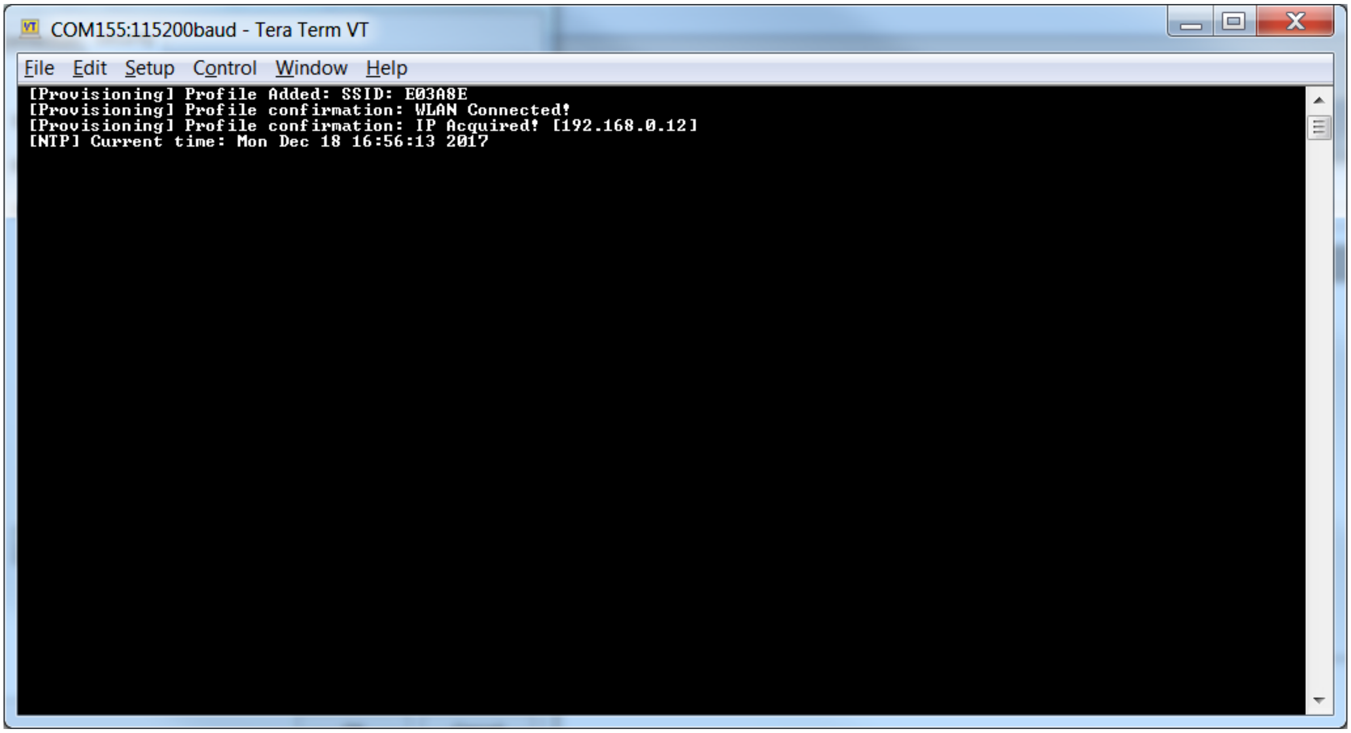
10.123.45.1 says:

1.Check command console and make sure device is connected
  and take note of the IP Address.
2.Connect to: NETGEAR
3.Type the device IP in the browser address bar.

**If Device provision failed, wait for the device network to show, and reconnect to it and try again.

Figure 76. Pop-up Message

6. Go to the serial terminal and make sure to save the new IP address assigned to the CC3220 gateway.

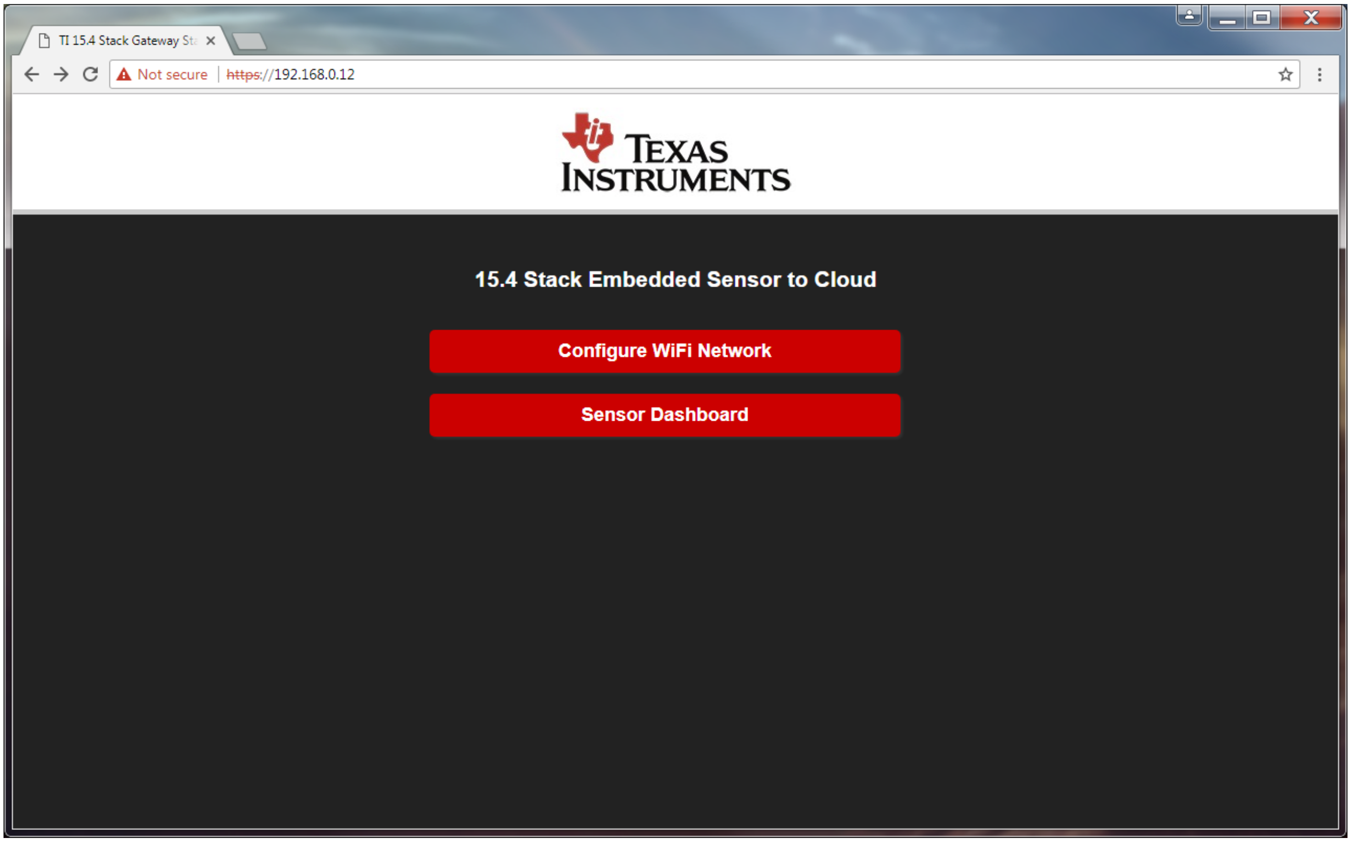


COM155:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
[Provisioning] Profile Added: SSID: E03A8E
[Provisioning] Profile confirmation: WLAN Connected!
[Provisioning] Profile confirmation: IP Acquired! [192.168.0.12]
[NTP] Current time: Mon Dec 18 16:56:13 2017
```

Figure 77. IP Address displayed on Serial Terminal

7. Using a computer or a mobile device connect to the same WiFi access point as the CC3220 gateway, open an internet browser and go to the IP addess assigned to the gateway, in this case it would be https://192.168.0.12 this should open the start page shown below.



15.4 Stack Embedded Sensor to Cloud

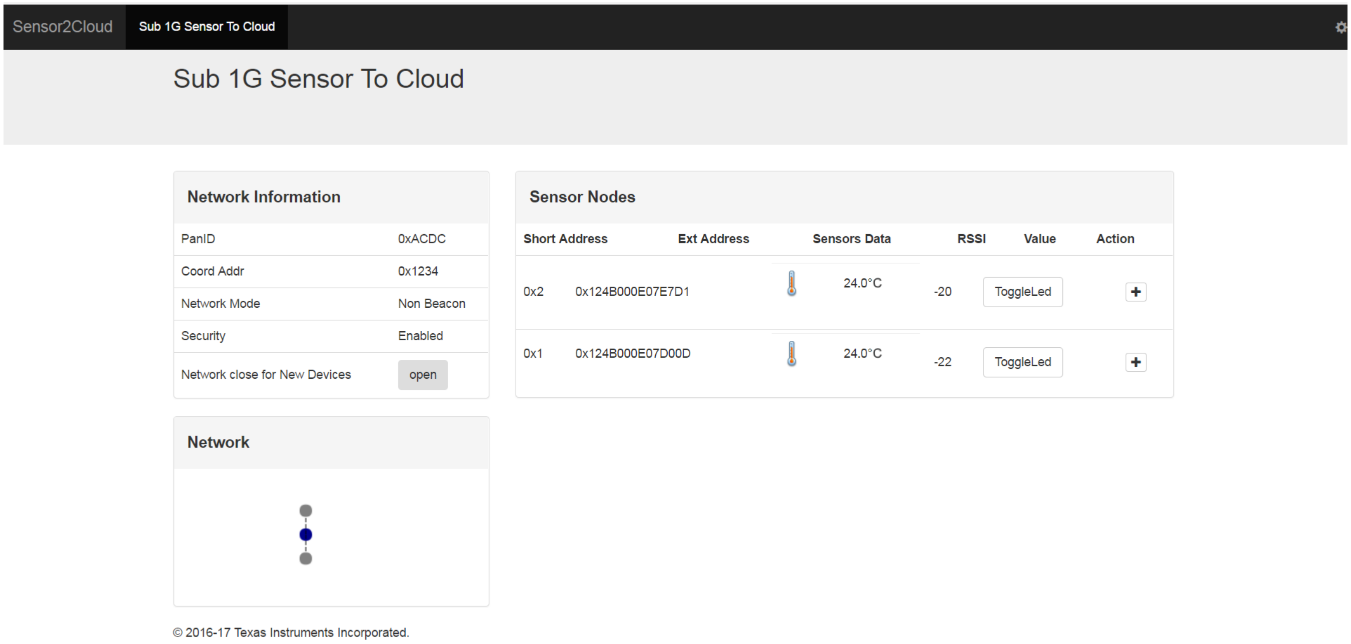
Figure 78. Start Page

8. Go to the section *Using SimpleLink Starter Pro App* at the beginning of Section 3.2.1.1.9, and follow the instructions starting from step 8.

3.2.2 Test Results

3.2.2.1 IoT Dashboard

Figure 79 shows is an example of the IoT dashboard displayed on the web interface. Note that the current network information is shown. The network chart displays the number of connected devices, and the sensor nodes section shows the device and current sensor information for all the devices in the network.



Sub 1G Sensor To Cloud

Network Information

PanID	0xACDC
Coord Addr	0x1234
Network Mode	Non Beacon
Security	Enabled
Network close for New Devices	open

Sensor Nodes

Short Address	Ext Address	Sensors Data	RSSI	Value	Action
0x2	0x124B000E07E7D1	🌡 24.0°C	-20	ToggleLed	+
0x1	0x124B000E07D00D	🌡 24.0°C	-22	ToggleLed	+

Network

© 2016-17 Texas Instruments Incorporated.

Figure 79. IBM Sensor-to-Cloud Dashboard Results

4 Design Files

This reference design showcases the connectivity between CC3220SF and CC13x0 devices. The CC3220SF acts as a gateway processor and the CC13x0 as communication node. The CC3220SF LaunchPad is used as a platform for gateway processor, and the CC13x0-based LaunchPad acts as communication node. The recommended schematics for this reference design uses the schematics of the CC3220SF LaunchPad and CC13x0 LaunchPad and interface the two devices using the UART lines. This IoT gateway reference design only uses one UART port. In addition, bootloader backdoor pins are described in the CC2538/CC26xx Serial Bootloader Interface application report. These pins can be connected to upgrade the firmware on the CC13x0 using the serial ROM bootloader on the CC13x0 devices.

4.1 Schematics

To download the schematics for this reference design, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

4.2 Bill of Materials

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

4.3 PCB Layout Recommendations

For layout prints, Altium project files, Gerber files, and assembly drawings, see the following links:

- LAUNCHXL-CC1310
- LAUNCHXL-CC1350
- CC3220SF-LAUNCHXL

5 Software Files

To download the software files, see the link at https://git.ti.com/tidc01002/tidc01002.

6 Related Documentation

1. Texas Instruments, SimpleLink TI 15.4-Stack IEEE 802.15.4e/g Standard Based Star Networking Software Development Kit, Tools Folder
2. Texas Instruments, SimpleLink CC3220 SDK, Tools Folder
3. Texas Instruments, TI 15.4-Stack Wiki, Wiki Page
4. Texas Instruments, *TI 15.4-Stack Embedded Developers Guide*

6.1 Trademarks

LaunchPad, Internet-on-a-chip, SmartConfig, Code Composer Studio are trademarks of Texas Instruments.
SimpleLink is a trademark of Texas Instruments Incorporated.
Cortex is a registered trademark of ARM Limited.
ARM, Cortex are registered trademarks of Arm Limited.
Bluetooth is a registered trademark of Bluetooth SIG, Incorporated.
IOS is a registered trademark of Cisco.
Android is a trademark of Google Inc.
IBM Watson IoT is a registered trademark of International Business Machines Corporation.
Linux is a registered trademark of Linus Torvalds.
Wi-Fi, Wi-Fi Direct are registered trademarks of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

7 About the Authors

HECTOR RAMOS is a software applications engineer at Texas Instruments, where he has been working at since 2014 and is specialized in low-power wireless protocols. Hector earned his bachelors of science in electrical engineering and computer science from University of California Berkeley.

ANDRES BLANCO is an applications engineer at Texas Instruments, where he is responsible for supporting customers designing low power wireless systems. Andres earned his bachelor of science in computer engineering from University of Puerto Rico in Mayagüez, PR.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (July 2017) to A Revision	**Page**
• Changed all instances of *TI Design* to *reference design*.	2
• Changed CC1310 to CC13x0 and CC1350 to CC13x02 throughout	2
• Changed cloud service provider to IBM Watson IoT	2
• Changed Software Block Diagram of TI 15.4-Stack Sensor-to-Cloud Reference Design image	4
• Added System Design Theory section and moved TI IoT Gateway-to-Cloud Service Interface section	7
• Added (for IBM Cloud only) to the Software section	11
• Changed CC3220 SDK v1.30.01.03 to CC3220 SDK v1.50.00.06 in the Software section	11
• Added SimpleLink CC13x2 SDK and removed v1.30 from SimpleLink CC13x0 SDK	11
• Added AWS IoT cloud services to Test Setup section	12
• Changed Running the Out-of-Box Examples section	12
• Added Running IBM Gateway Example section and changed Bluemix to Cloud	12
• Added Running the AWS IoT Example section	13
• Changed section title from *Program the MAC CoP LaunchPad™* to *Programming the CC13x0 LaunchPad™*	13
• Changed MAC CoP to CC13x0 in the Programming the CC13x0 LaunchPad™ section	13
• Changed CC1350 LaunchPad CoProcessor.hex or sensor_cc1350lp_doorlock.hex to CoProcessor or Sensor hex files provided in the prebuilt directory in the Programming the CC13x0 LaunchPad™ section	14
• Changed out-of-the-box demonstration to CC3220SF LaunchPad in the Programming the CC3220SF LaunchPad™ section	15
• Changed directory path from C:\<S2C_Repo_Directory>\CC3220_CC13x0Gateway\prebuilt\CC3220SF_LaunchXL\Uniflash_CC3220ImageCreatorProject.zip to C:\<S2C_Repo_Directory>\tidc01002\prebuilt\CC3220SF_LaunchXL\Uniflash_CC3220ImageCreatorProject.zip in Programming a Preconfigured Image Creator Project section	18
• Added Importing Examples to Code Composer Studio (CCS) section	21
• Added Compiling the Mac-CoP from Source section	23
• Changed trusted root-certificate catalog directory path in the Creating an Image Creator Project in UniFlash section	26
• Changed service pack directory path in the Creating an Image Creator Project in UniFlash section	27
• Changed directory path from C:\<S2C Repo Directory>\\CC3220_CC13x0Gateway\\examples\\CC3220SF_LaunchXLcc3220sf_gateway_app/15_4_Stack_Gtway_CC3220SF_LAUNCHXL_tirtos_ccs/ to C:\<S2C Repo Directory>tidc01002\src\www in the Creating an Image Creator Project in UniFlash section	28
• Added instructions for the www folder in the Creating an Image Creator Project in UniFlash section	28
• Changed section title to Connecting the MAC-CoP and the CC3220SF LaunchPad™ and updated jumper/configuration instructions	34
• Changed content in Open and Configure an IBM Cloud™ Account section	36
• Changed content in Set Up Watson IoT™ Platform Service section	41
• Changed content in Set Up Node.js Cloud Foundry App section	49
• Changed cd C:\<S2C Repo Directory>\CC3220_CC13x0Gateway\\examples/ibm_cloud_application to cd C:\<S2C Repo Directory>\tidc01002\examples\ibm_cloud_application in Set Up Node.js Cloud Foundry App section	50
• Changed content in Using the SimpleLink™ Starter Pro App section	59
• Changed content in Using the Local Provisioning Web Page section	69

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated (‘TI”) technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, “TI Resources”) are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI’s provision of TI Resources does not expand or otherwise alter TI’s applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED “AS IS” AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI’s standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265