Getting Started Guide: Matter over Wi-Fi on the CC32XX

Contents

1. Overview

This document provides instructions on how to get started with developing Matter Applications on the CC3235SF device.

The CC32XX currently supports the lock-app example demonstrating the use of OnOff cluster (see in *{connectedhomeip-root}/examples/lock-app/cc32xx/*). The Lock/Unlock operations are emulated through the status of green LED on the launchpad.

The CC32XX currently needs to be on the target network before running the Matter protocol (using the Matter's "OnNetwork" configuration). When running the example for the first-time, provision the device using the Simplelink StarterPro mobile application or use hard-coded settings of the SSID and password in "main/wifi-settings.h". The status of the Wi-Fi is represented through the Launchpad's blue LED (on - when connected, blinking – while provisioning).

The CHIP Tool is an emulation of a Matter controller that can be used to test the CC32xx device. There are several variations of the tool for different environments and each has different behaviors (as not all the Matter's features are supported currently). The CC32xx was tested with against the following CHIP Tools:

- C++ CHIP Tool (in *{connectedhomeip-root}/examples/chip-tool/*) – Linux command line application (each command following the pairing creates a secure CASE session to the device before send the control command to the device). The C++ CHIP Tool support device discovery through MDNS.
- Python CHIP Tool (in *{connectedhomeip-root}/src/controller/python/*) – In this Python shell, the secure session is opened once at the pairing and next commands are executed immediately in the secure context.
- Android CHIP Tool (In *{connectedhomeip-root}/src/android/CHIPTool*/) – Android Test application for mobile device

2. Building the Examples

2.1 CC32xx Lock App Demo

The README for the Lock App Demo details how to build the example for the CC32XX.

Before programming the lock app demo onto the device, the Service Pack for the CC32XX needs to be programmed via UniFlash. Connect the device to your laptop, choose the CC3235SF-LAUNCHXL as the device, and start the Image Creator.

Click “Start new project”, enter the project name, device type, and set Device mode to “Develop” and click “Create Project”.

Click on the Advanced view. Under User Files, add the dummy-root-ca-cert (*{sdk_root}/tools/cc32xx_tools/certificate-playground)* and under Service Pack, add the .bin file from the SDK (*{sdk_root}/tools/cc32xx_tools/servicepack-cc3x35).* You can now connect to the device and program the ServicePack.

3. Building the CHIP Tool

3.1 C++ CHIP Tool

The README for the C++ CHIP tool details how to build the project. One change that needs to be made is in ModelCommand.h – the timeout (line 44) needs to be increased from 10 to 15 seconds. Once this change is made, the example can be built.

3.2 Python CHIP Tool

The README for the Python CHIP Tool details how to build the project. For Step 5, instead of using the command listed, run “scripts/build_python.sh -i separate” to get the correct files needed to run the CHIP tool.

3.3 Android CHIP tool

The README for the Android CHIP details how to build the project.

Installing the app on a mobile device can be done through ADB (see the README) or by opening the APK file on the mobile device.

4. Running the examples

4.1 Lock App Demo

The device needs to be provisioned onto the local network before it can be commissioned onto a Fabric. Once the example has been built and is running on the CC3235SF LaunchPad, provision the device using the Simplelink Wi-Fi Starter Pro App. To learn how to provision a device onto a network, please refer to the following lab from Simplelink Academy: https://dev.ti.com/tirex/explore/node?node=AIlUqB-y4S3M7c6JO9hsuQ__fc2e6sr__LATEST

Once the device is on the network, the device can be commissioned by the CHIP tool.

Important Note: the device (Linux machine or Android Phone) running the CHIP Tool must be connected to the same local network the CC3232SF was provisioned to.

4.2 C++ CHIP Tool

To commission the device onto the Fabric, run the following command:

>> chip-tool pairing onnetwork <nodeId> 20202021

Once the device is commissioned, run the following command to lock:

>> chip-tool onoff on <nodeId> 1

This command can be run to unlock:

>> chip-tool onoff off <nodeId> 1

The C++ CHIP tool can also be used to send commands to a commissioned device once it has been reset. The following commands can be used:

>> chip-tool pairing onnetwork <nodeId> 20202021

<reset the CC32XX device>

>> chip-tool basic read node-label <nodeId> 0

>> chip-tool onoff on <nodeId> 1

>> chip-tool onoff off <nodeId> 1

4.3 Python CHIP Tool

To commission the device onto the Fabric, run the following command:

>> connect -ip <address (the embedded device's IP address is provided after the device is provisioned) > 20202021 <nodeId>

Alternatively commissioning can be triggered using the QR code (the QR code can be found in the terminal log during device init):

>> connect -ip <address (the embedded device's IP address is provided after the device is provisioned) > 20202021 <nodeId>

>> connect -qr <QR-Code>

By default the QR code is: MT:8IXS1AFN00KA0648G00

To lock and unlock the device, run the following commands:

>> zcl OnOff On <nodeId> 1 0

>> zcl OnOff Off <nodeId> 1 0

4.4 Android CHIP Tool

To commission the device onto the Fabric, run the following command:

Click the “PROVISION CHIP DEVICE WITH WI-FI” and then click “INPUT DEVICE ADDRESS” and provide the device’s IP address (as printed to the terminal log). Finally click the “COMMISSION” button to trigger the commissioning sequence.

Note: QR scanning through the app is currently working only for BLE commissioning (which is not supported by CC32XX).

Use “LIGHT ON/OFF & LEVEL CLUSTER” to control the LED on the launchpad.