EasyEVSE Entry-level Development Platform with Matter and OCPP

Hitesh Mathukiya, Sri Anand Pochiraju, Alexandru Sava

Technical Training – May 2026

| **Public** | NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2026 NXP B.V. Version 3.0.

NXP Public



Agenda

- NXP EasyEVSE Entry-level development platform – system overview
- Hardware building blocks
  - RW612 host controller
  - KM35Z75M metrology MCU
  - Solution setup
- Software Building Blocks
  - Charging service (Metrology and IEC61851)
  - Open Charge Point Protocol (OCPP communication and application)
  - EVSE Matter and application
- Demonstration video
- Q&A

Public

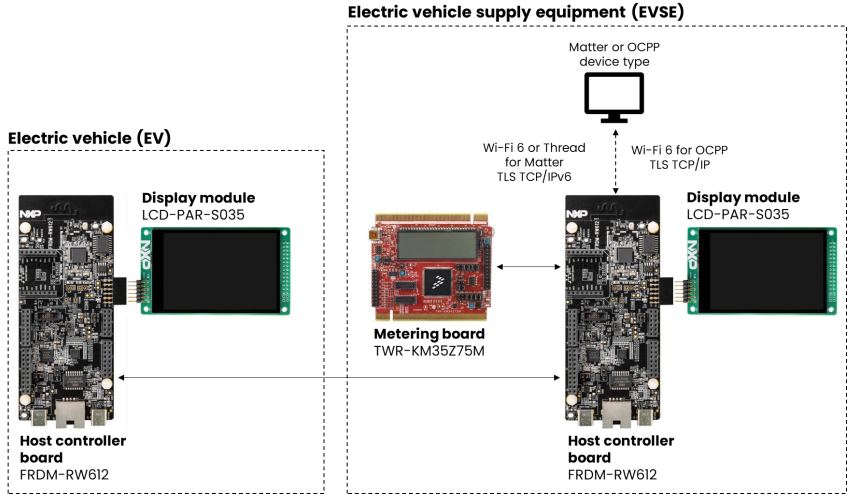
EasyEVSE Entry-level Development Platform (RW612 MCU, FreeRTOS, Wi-Fi 6)

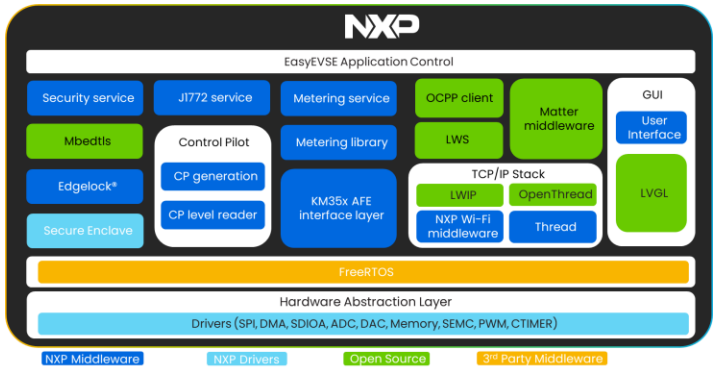
NXP accelerates fast, differentiated entry-level single- and multi-phase AC EV charger development

- **Reduce design, complexity, cost** and **footprint** by converging **4** chips into **1**
- Benefit from industry-leading **1x1 dual-band Wi-Fi 6** for **extended range**, **higher data throughput**, and **robust connectivity**
- **Matter-enabled** EV charging applications over **Wi-Fi 6** or **Thread**
- Ensure seamless communication between EVSE and cloud services using the proven **open-source EVeret OCPP 2.0.1** implementation
- Enable **proven resilient EdgeLock Secure Enclave** for secure backend communications
- Achieve accurate, reliable billing and advanced power-line monitoring using **modifiable, pre-certified metrology** software
- Enable **application software reuse** to scale up to higher-performance EasyEVSE MCU and MPU platforms

10+ 15+
Product Longevity

Solution info: click here





Get started
Application note and Get Started instructions

Development hardware
Order from nxp.com

Complementary development software
Download from github.com/NXP

Solution order numbers

FRDM-RW612*	Host Controller Board
TWR-KM35Z75M	Metering Board
LCD-PAR-S035*	3.5" 480x320 IPS TFT LCD Module

*Choose one for the EV and one for the EVSE side

NXP components

RW612	RW612 Wi-Fi/BLE MCU
MKM35Z75M	KM35 metrology MCU

NXP Public

Overview

- **Objective**: Develop a secure Electric Vehicle Supply Equipment (EVSE) reference design compliant with Matter/OCPP standards.

- **Platforms**:
  - **FRDM-RW612** → Hosts Matter stack, charging control, metrology, and secure onboarding.
  - **TWR-KM35Z75M** → Provides precision metering and energy session reporting.

- **Key Features**:
  - Secure enclave with EL2GO credentials for hardened keys and onboarding.
  - FreeRTOS-based application supporting OCPP over Wi-Fi.
  - FreeRTOS-based application supporting Matter over Wi-Fi and Thread.
  - Accurate metering, charging state machine, and fault handling.

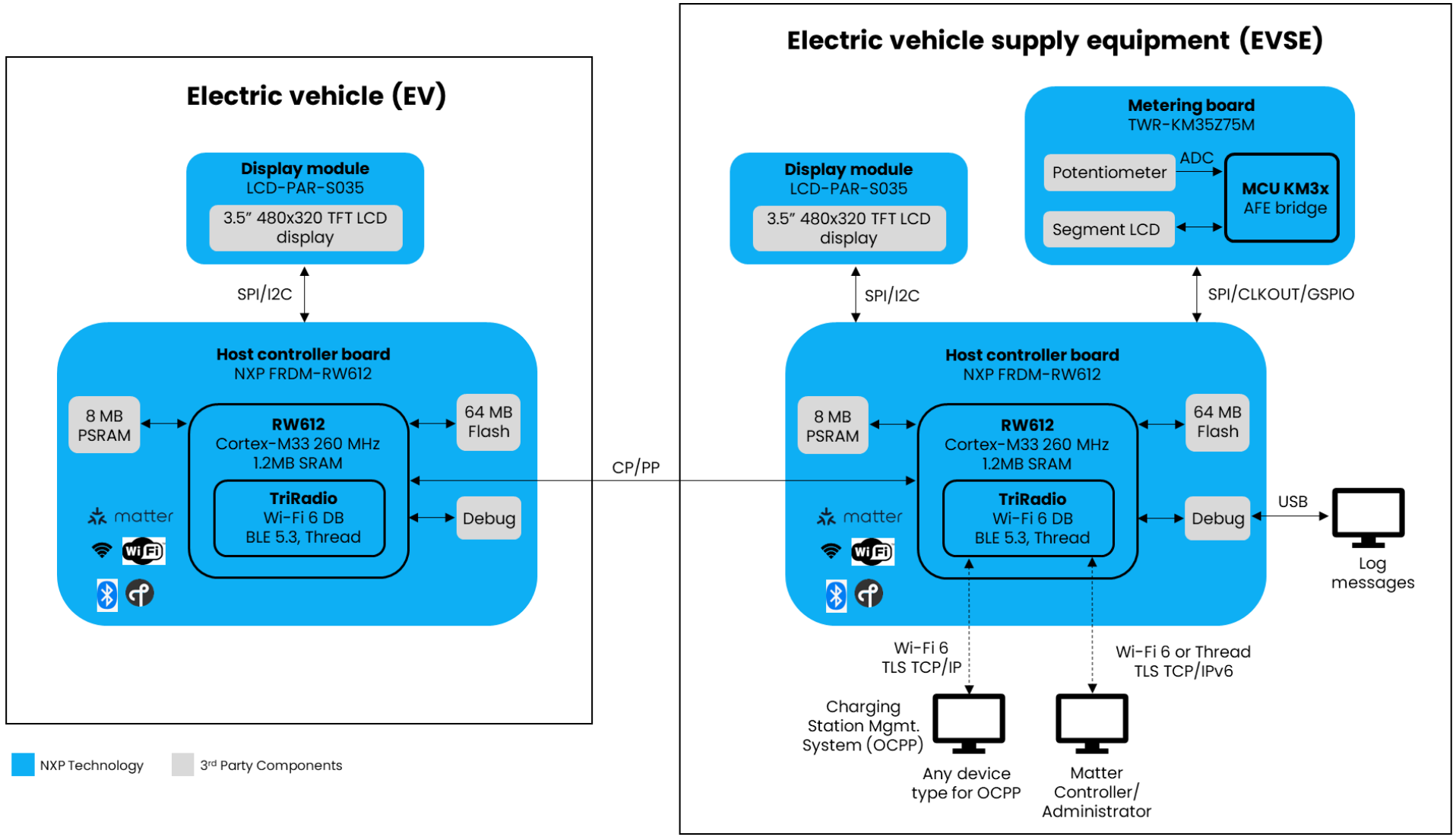
NXP Public

02

Hardware building blocks

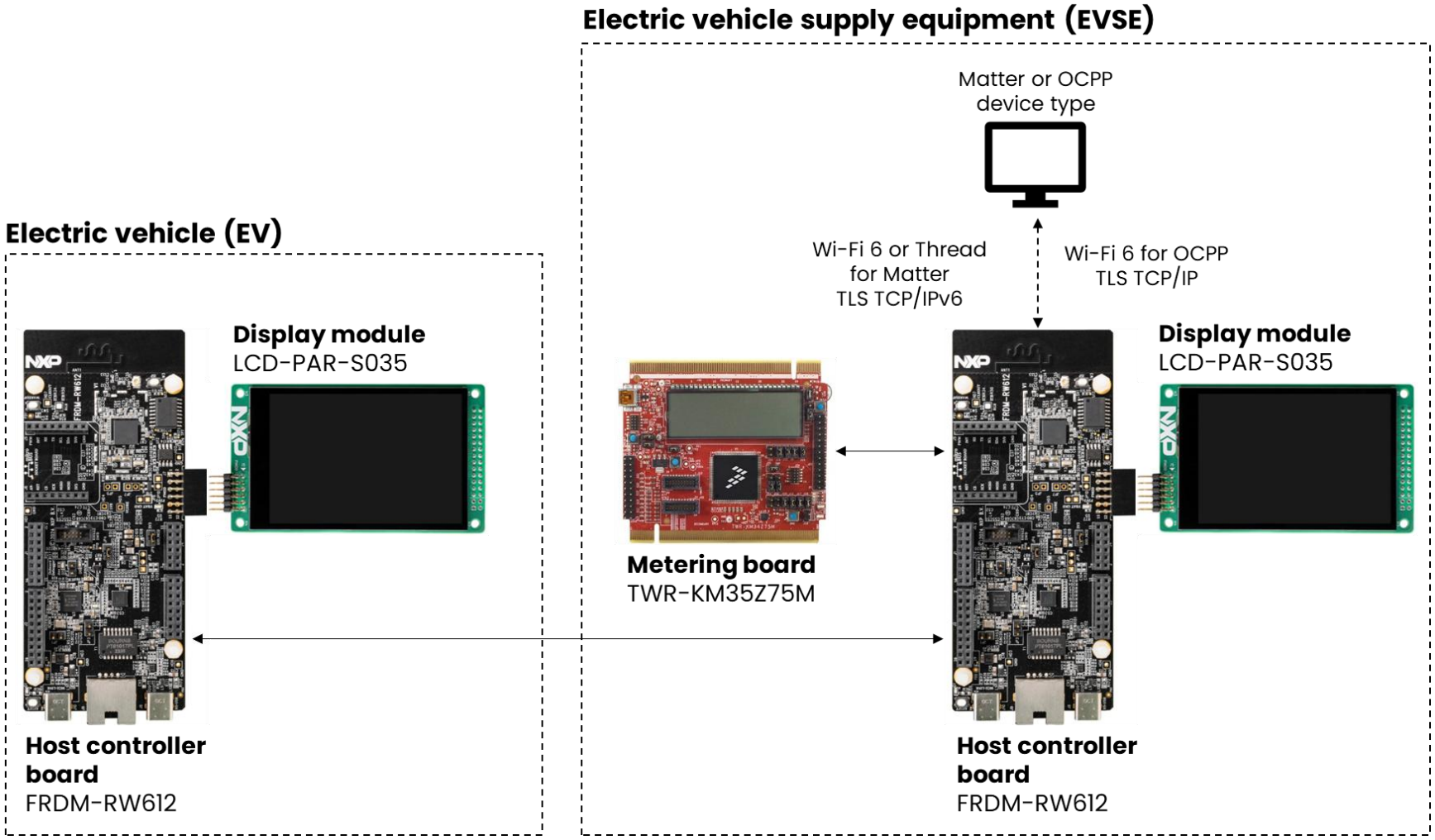


EasyEVSE Entry-level Development Platform – system block



NXP Public

EasyEVSE Development Platform – architecture



NXP Public

RW612

I/O Interfaces

- 5 x FlexComm UART, SPI, I²C, I²S/PCM
- Ethernet RMII
- SDIO 3.0
- HS USB 2.0 OTG
- LCD Interface
- 4 x Digital Mic.
- GPIOs (Up to 64)

Memory Interface

(Quad) FlexSPI
FLASH and PSRAM Interface

Power Management

Input: (1.8 V – 3.6 V)

Analog

- 2 x 16-bit ADC
- 10-bit DAC
- ACMP

Other Security Modules

- FLASH & PSRAM Encryption
- EdgeLock Accelerator PKC
- PUF

Dedicated Application CPU Platform

Core

- Arm®Cortex ®-M33 (260 MHz)
- PowerQuad Co-Processor
- FPU | MPU | NVIC
- TrustZone-M

Memory

1.2 MB SRAM	256 kB ROM
16 kB AON RAM	OTP Fuses

System Control

Secure Debug	Timers / PWM
IRC / OSC	DMA
RTC	Watch Dog

Radio Sub-System

Wi-Fi®Radio

Wi-Fi CPU Cortex-M3 160 MHz	Wi-Fi 6 1x1 20 MHz MAC/BB	5 GHz RF&SPDT
900 kB RAM		2.4 GHz RF&SPDT

NarrowBand (NB) Radio

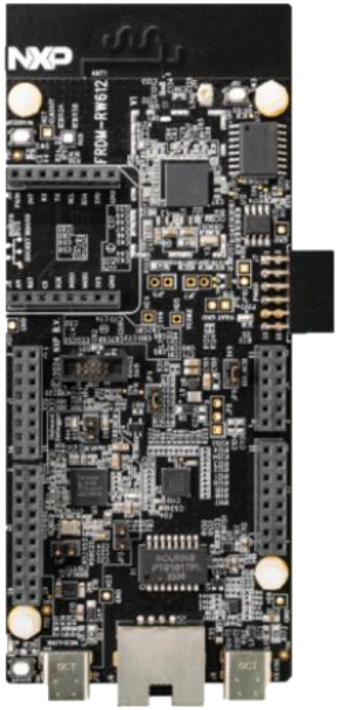
NB CPU Cortex-M3 128 MHz	BT LE Link Layer	2.4 GHz LNA / PA
464 kB RAM	802.15.4 MAC	SPDT

Multi-Radio Coexistence Arbiter

3 Internal Radios
1 External Radio

EdgeLock®Secure Enclave (Core Profile)

Security Core	Crypto Accelerators	TRNG
	Key Store	



NXP Public

KM35Z75M

Core

- ARM®Cortex ®-M0+ Up to 75 MHz
- SWD Interface
- MTB
- Interrupt Controller
- H/W Arithmetic Unit (Optional)

System

- Watchdog
- BME
- Low-Leakage Wake-Up Unit
- Inter Peripheral Crossbar
- EWM
- MPU

Memory

- 64-512 KB Flash
- 16-64 KB SRAM

Measurement Front End

- Up to 3 x HSCMP
- 4 x 24-bit Sigma Delta ADC
- 16-ch., 16-bit SAR ADC (Optional)
- Up to 4 x PGA
- 1.2 V VREF
- Phase Shifter

Security

- 16/32 Cyclic Redundancy Check
- MMCAU (Optional)
- Random Number Generator
- Tamper Detection

Display

- Segment LCD Controller (Up to 448 Segments, 56 x 8) (Optional)

Timers

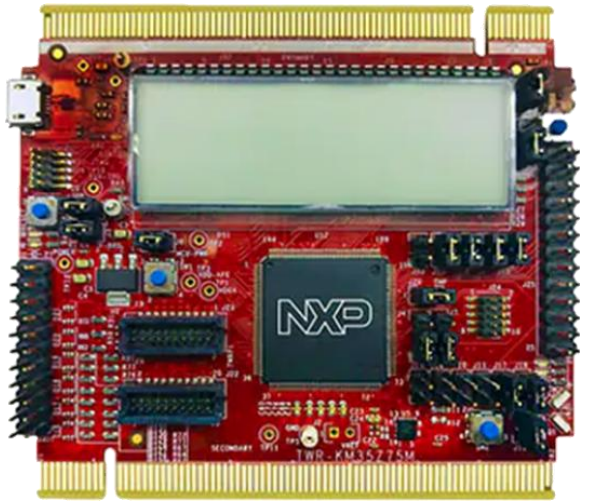
- 4 x Quad Timer
- Up to 2 x Low-Power Timer
- 2 x Periodic Interrupt Timers
- IRTC on vBatt with T° comp.
- Programmable Delay Block (Optional)

Communication Interfaces

- Up to 3 x 16-bit SPI
- 2 x I ²C
- Up to 5 x UARTs (2 x ISO7816, 1 x LPUART)
- Up to 99 GPIO

Clocks

- Phase Locked Loop
- Frequency Locked Loop
- Low/High Frequency Oscillators
- Internal Reference Clocks



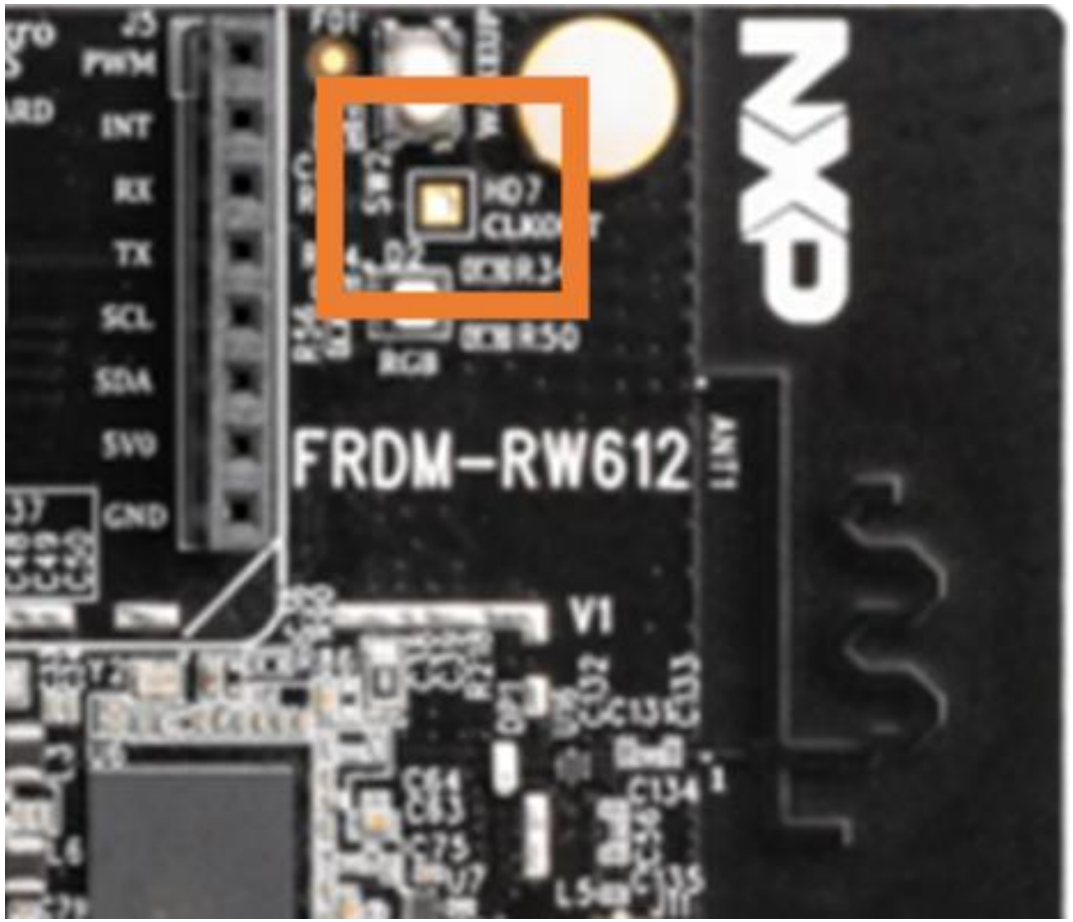
NXP Public

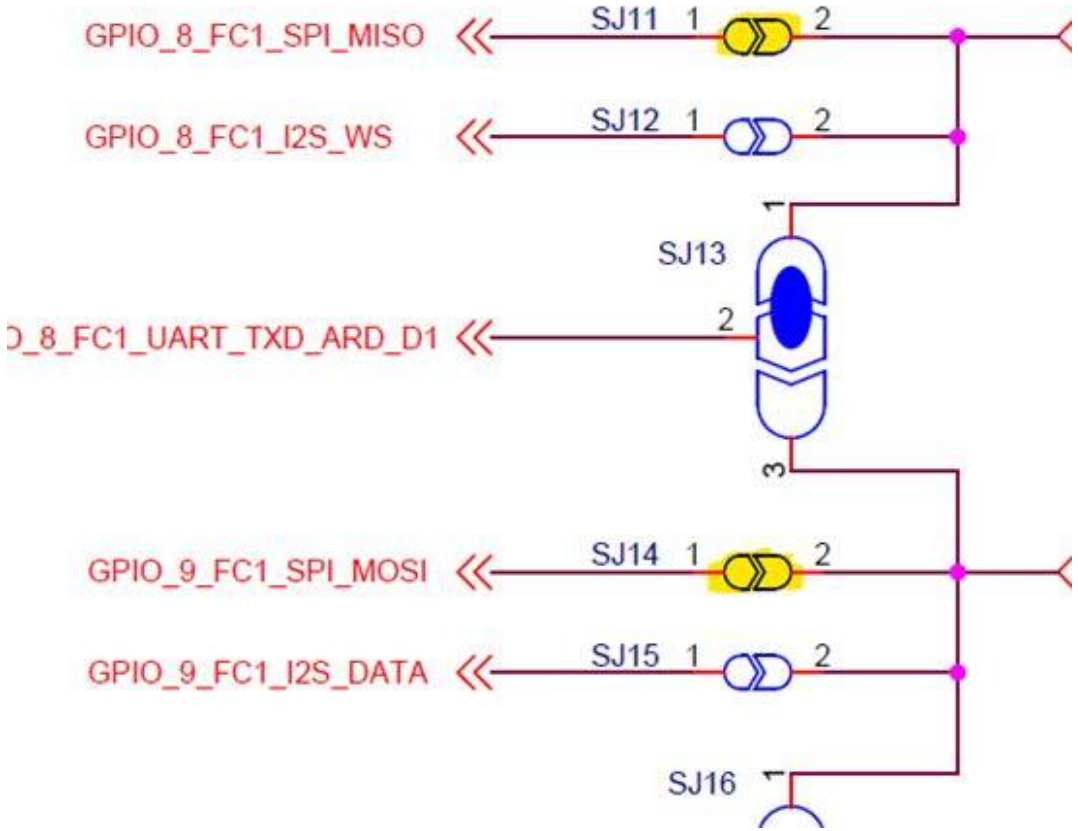
03

EVSE system setup

System assembly

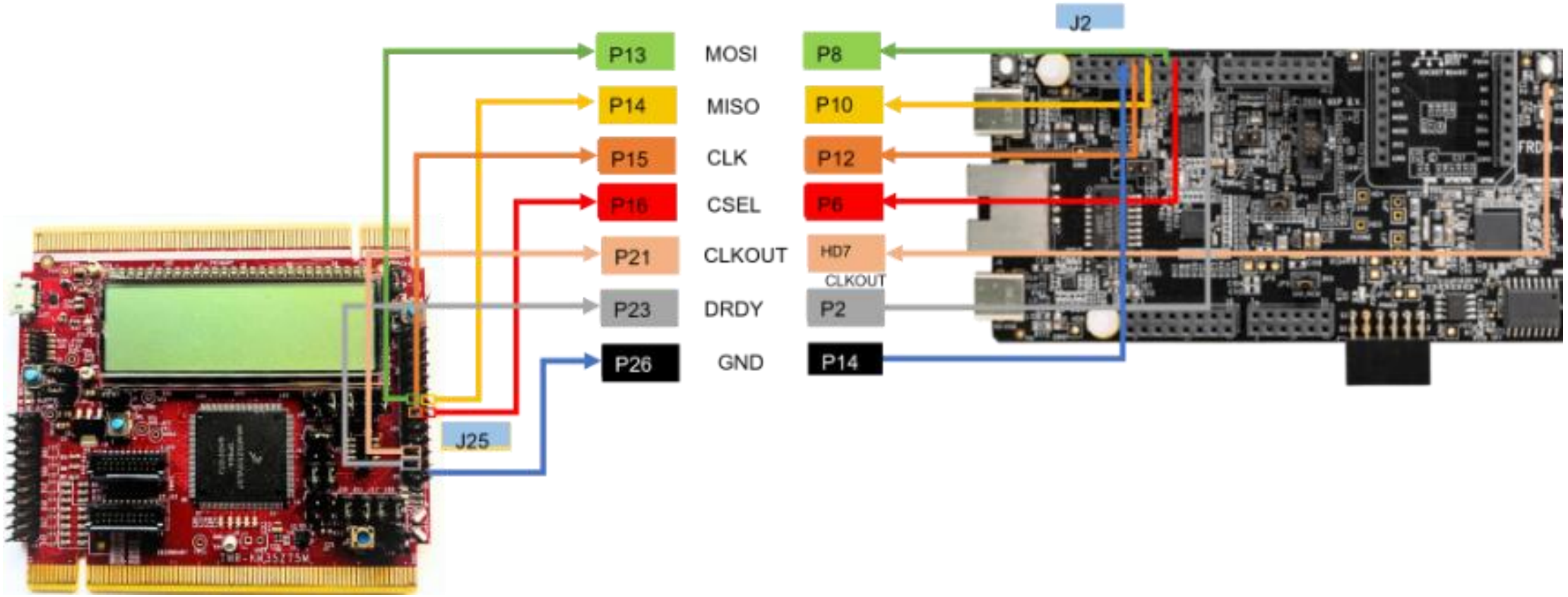
EVSE system setup - hardware modification





NXP Public

EVSE system setup – KM35x – EVSE



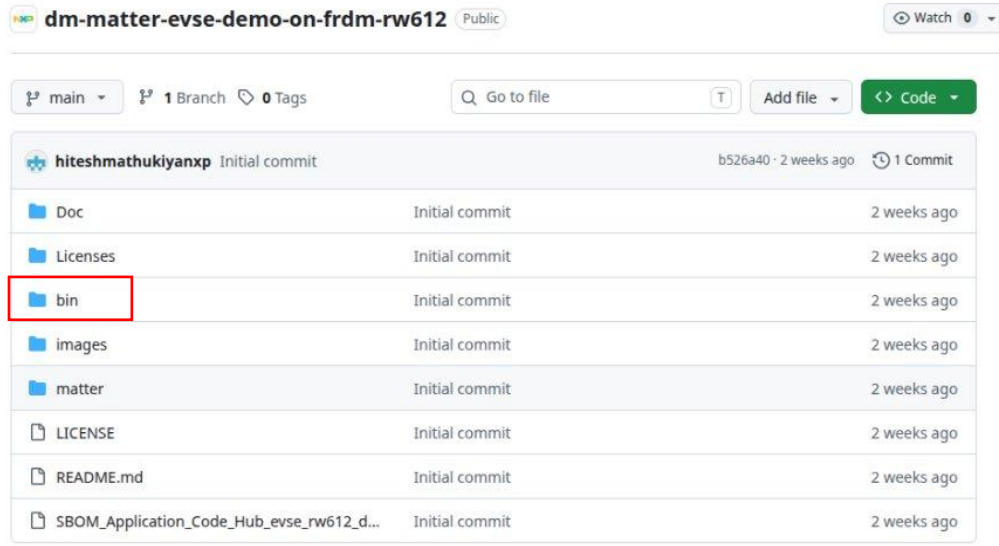
NXP Public

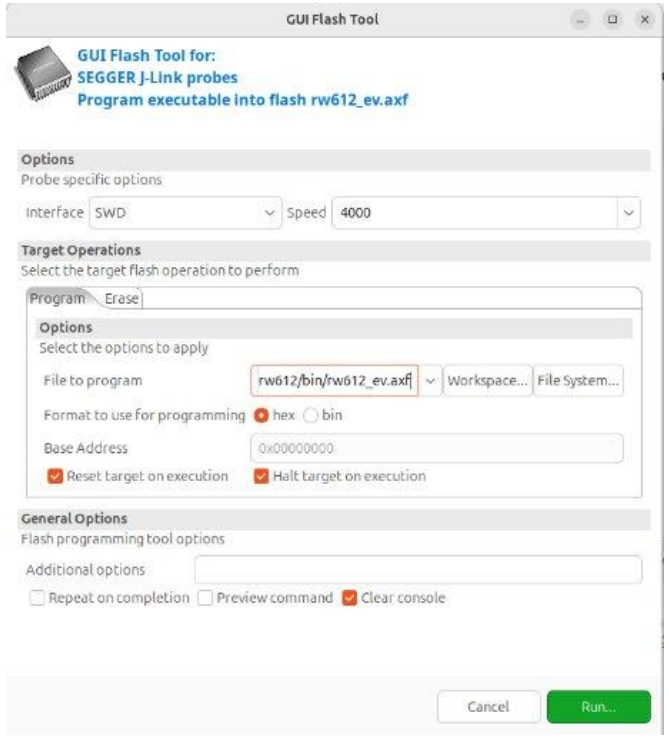
EVSE system setup – KM35x – EVSE

https://github.com/nxp-appcodehub/dm-nxp-km-metrology
https://github.com/nxp-appcodehub/dm-matter-evse-demo-on-frdm-rw612

Software setup and flashing

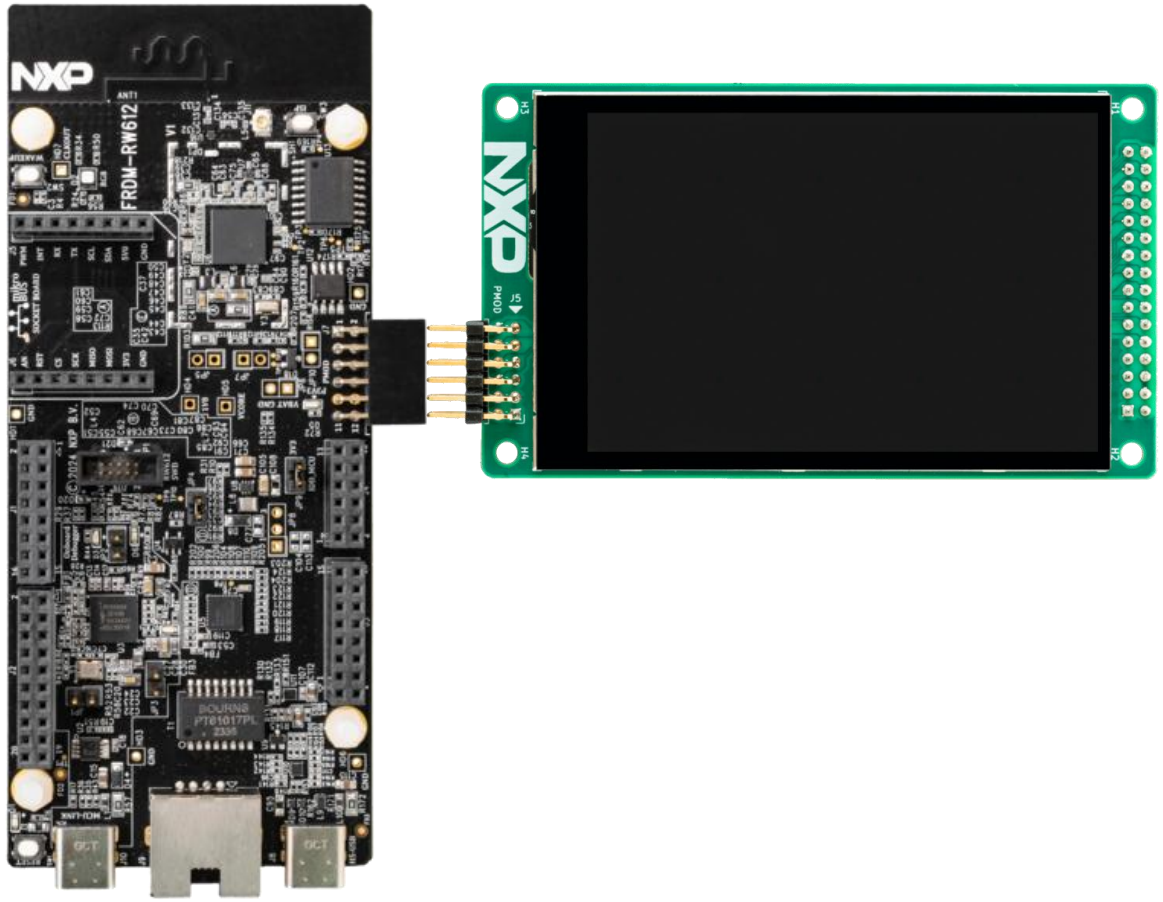
- Follow the README.md from the dm-nxp-km-metrology ACH
OR
- Flash using the .axf provided in the dm-matter-evse-demo-on-frdm-rw612 ACH





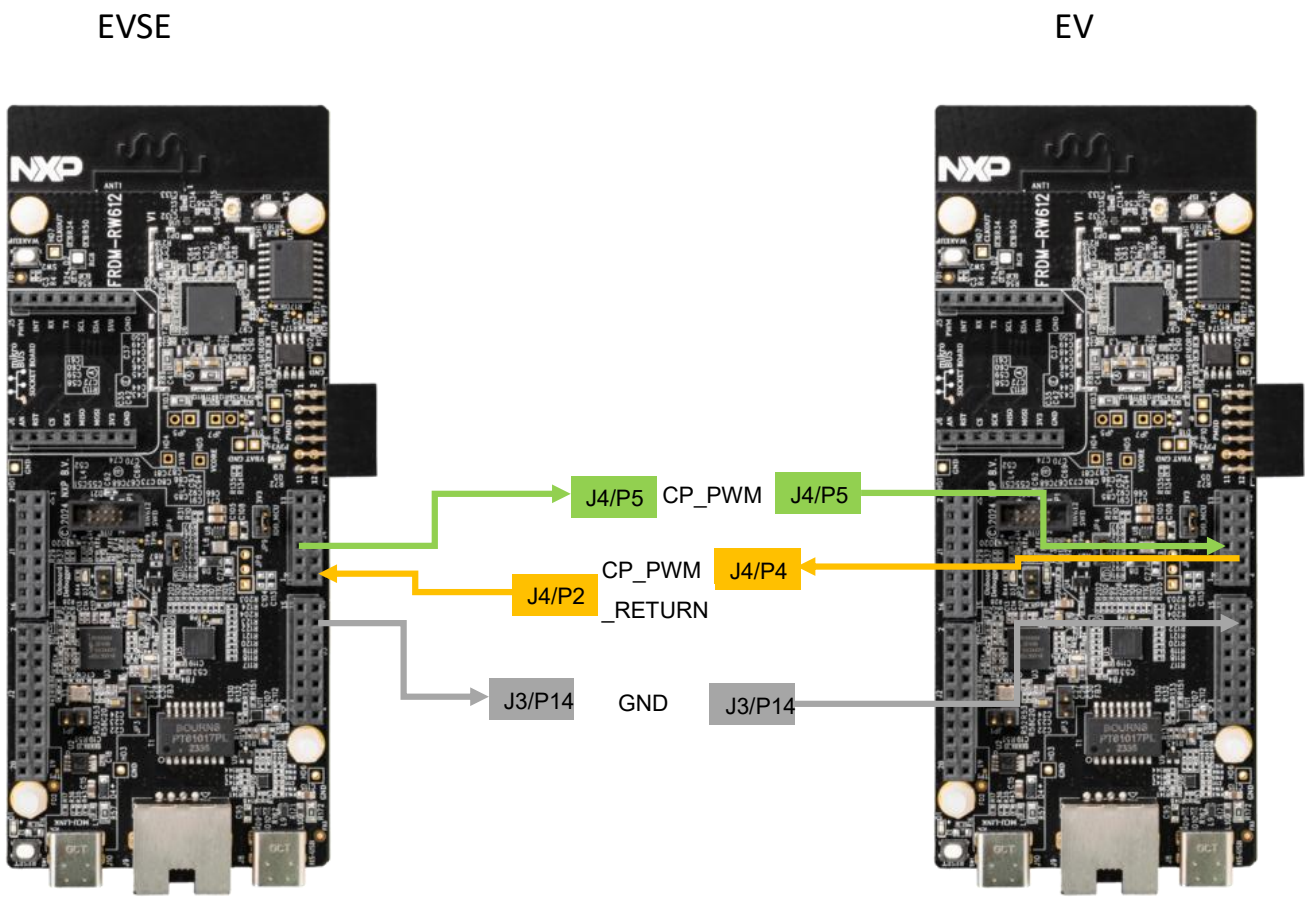
NXP Public

EV/EVSE system setup – display connection



NXP Public

EV/EVSE system setup



NXP Public

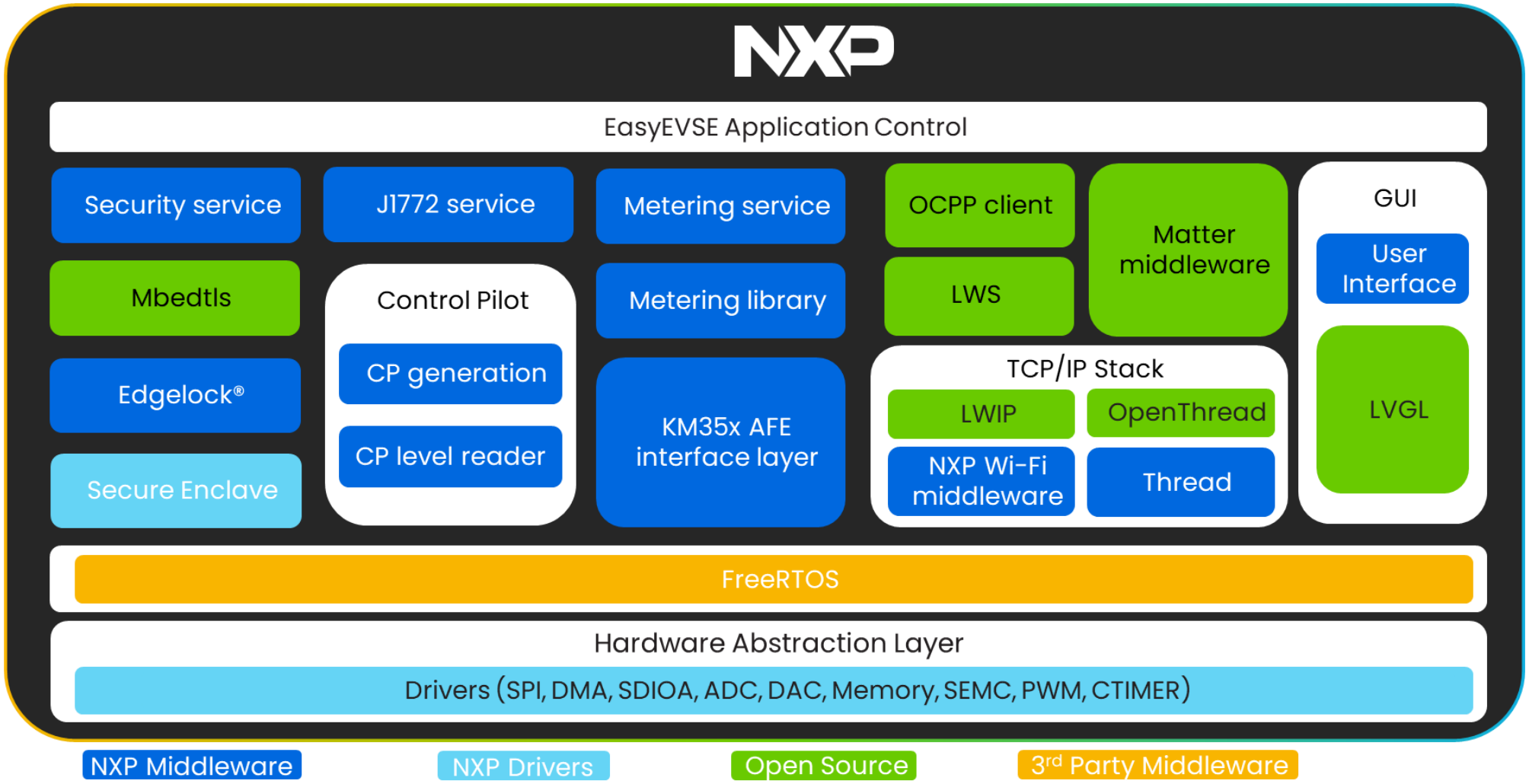
04

EasyEVSE Entry-level Development Platform

Software building blocks



Software block diagram



NXP Public

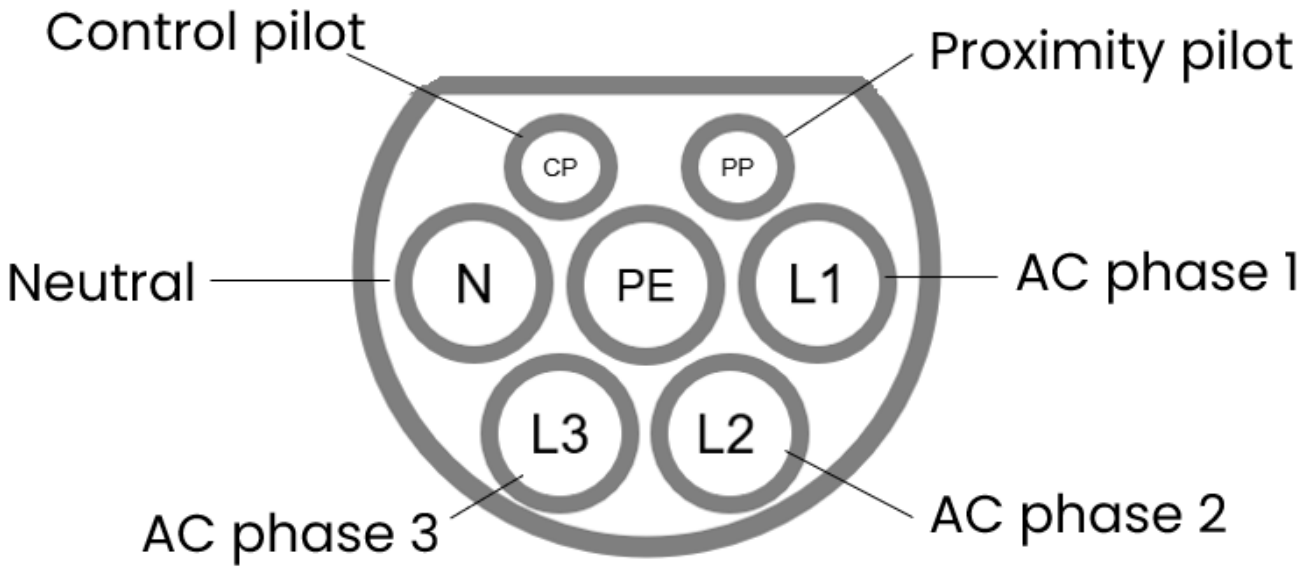
05

Charging service

Metrology and IEC61851



SAEJ1772/ IEC 61581-1



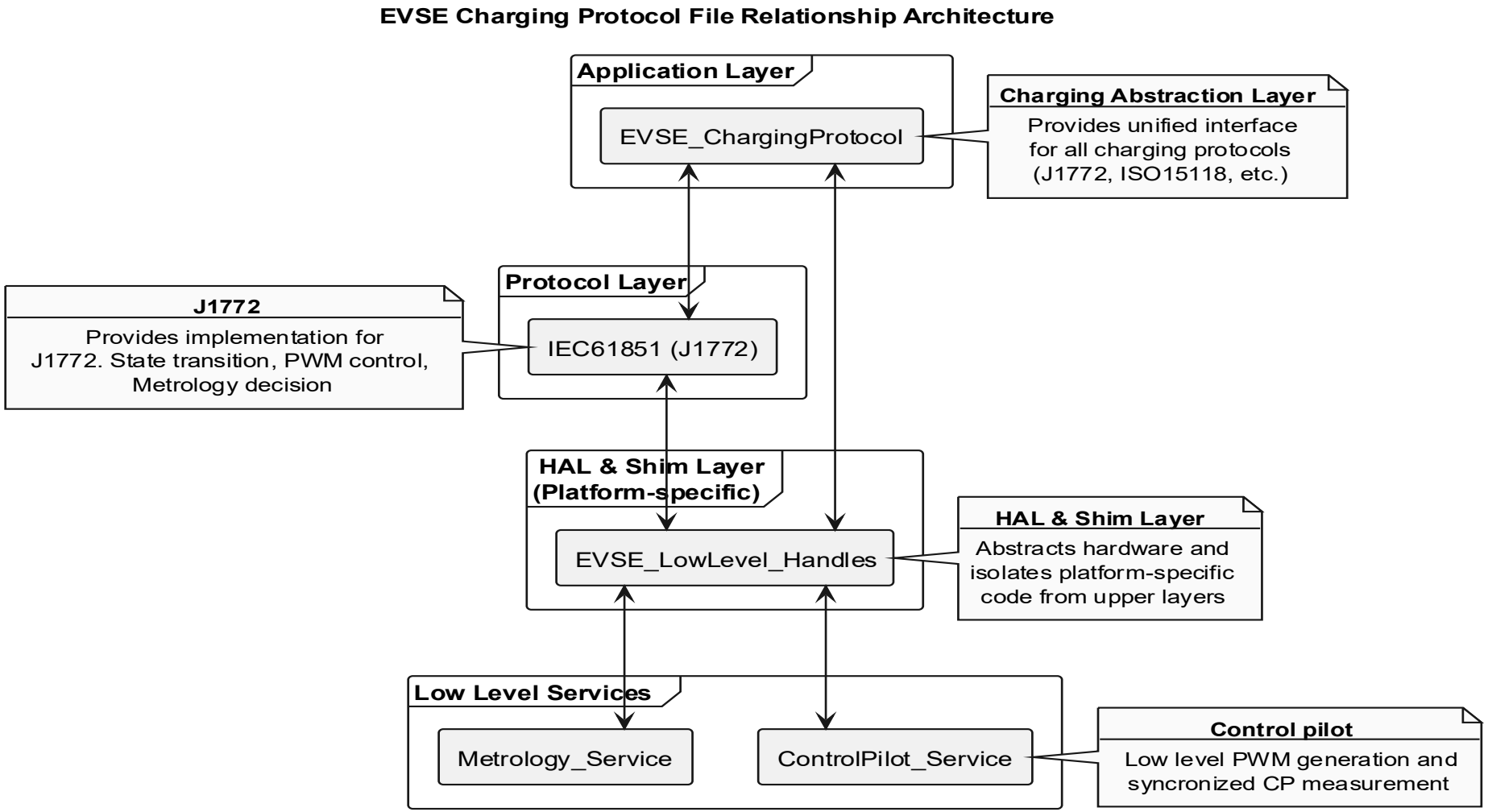
AC Charging connector type 2

Status	Charging state	Voltage max
State A	Standby	+12V
State B	Vehicle detected	+9 V
State C	Vehicle Ready	+6V
State D	Ready with ventilation	+3V
State E	No power	0V
State F	Error	-12V

Amps	Duty Cycle
6A	10%
18A	30%
65A	90%
80A	96%
6A-51A : Amps = Duty Cycle x 0.6 51A-80A: Amps = (Duty Cycle – 64) x 2.5	

NXP Public

Charging service architecture



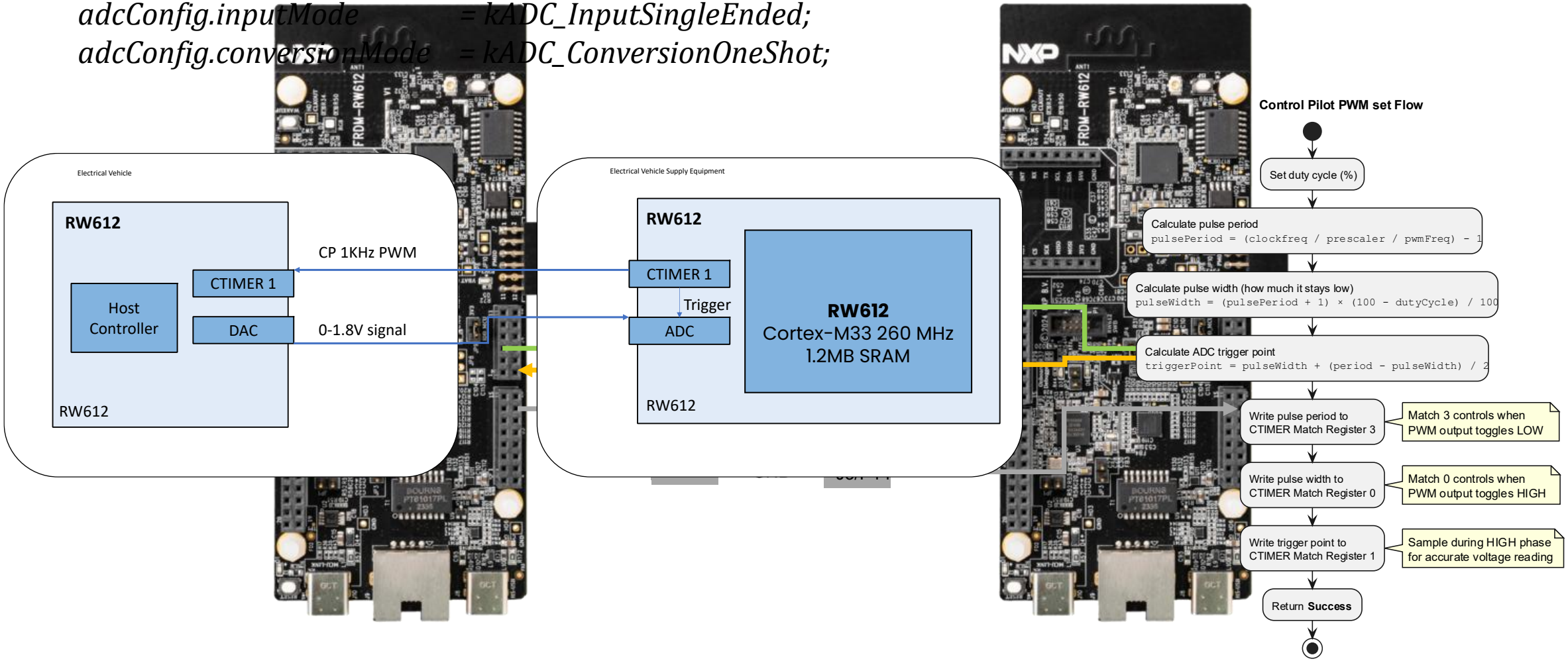
NXP Public

Control Pilot

adcConfig.resolution = kADC_Resolution16Bit;
adcConfig.averageLength = kADC_AverageNone;
adcConfig.inputMode = kADC_InputSingleEnded;
adcConfig.conversionMode = kADC_ConversionOneShot;

EVSE

EV



NXP Public

Metrology

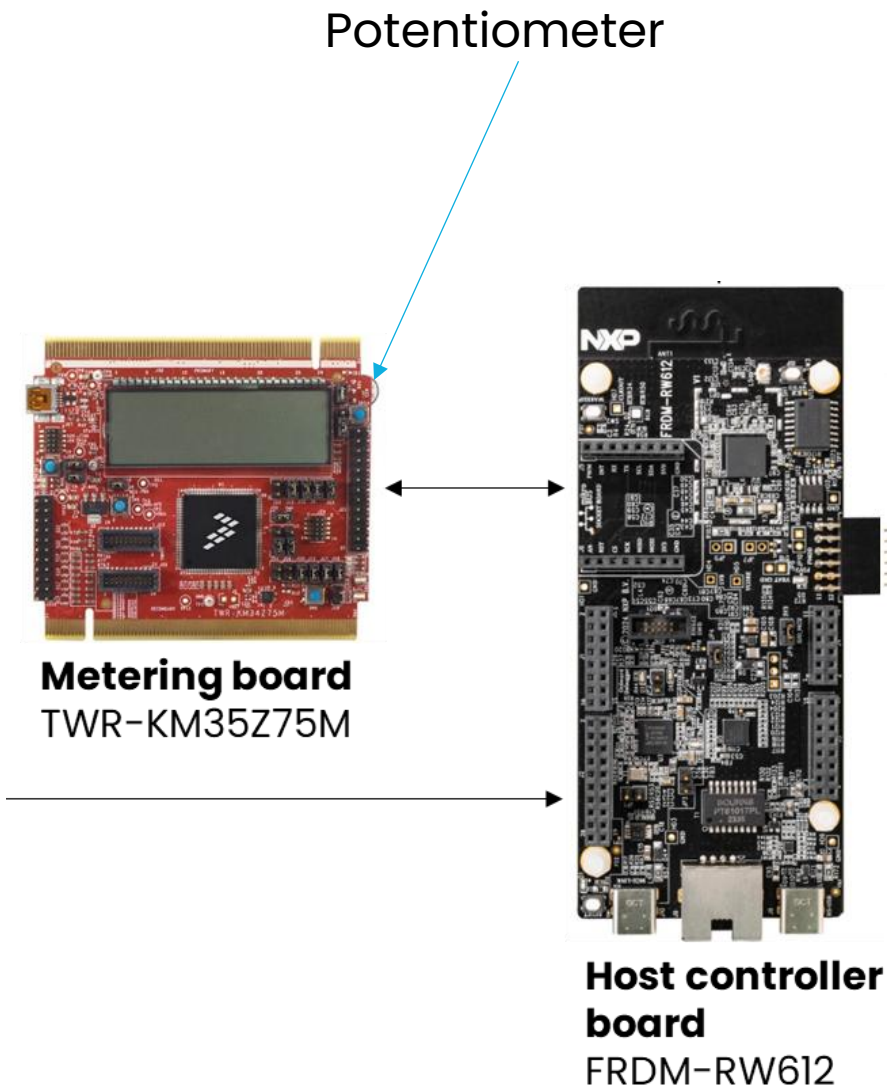
What does the KM35x do?

- Enables and configures the AFE
- Calculates simulated data based on potentiometer value
- Receives data from the AFE but can/will replace real measurements with simulated data
- Transmits the simulated metrology data to the RW612

❌ What doesn't the KM35 do?

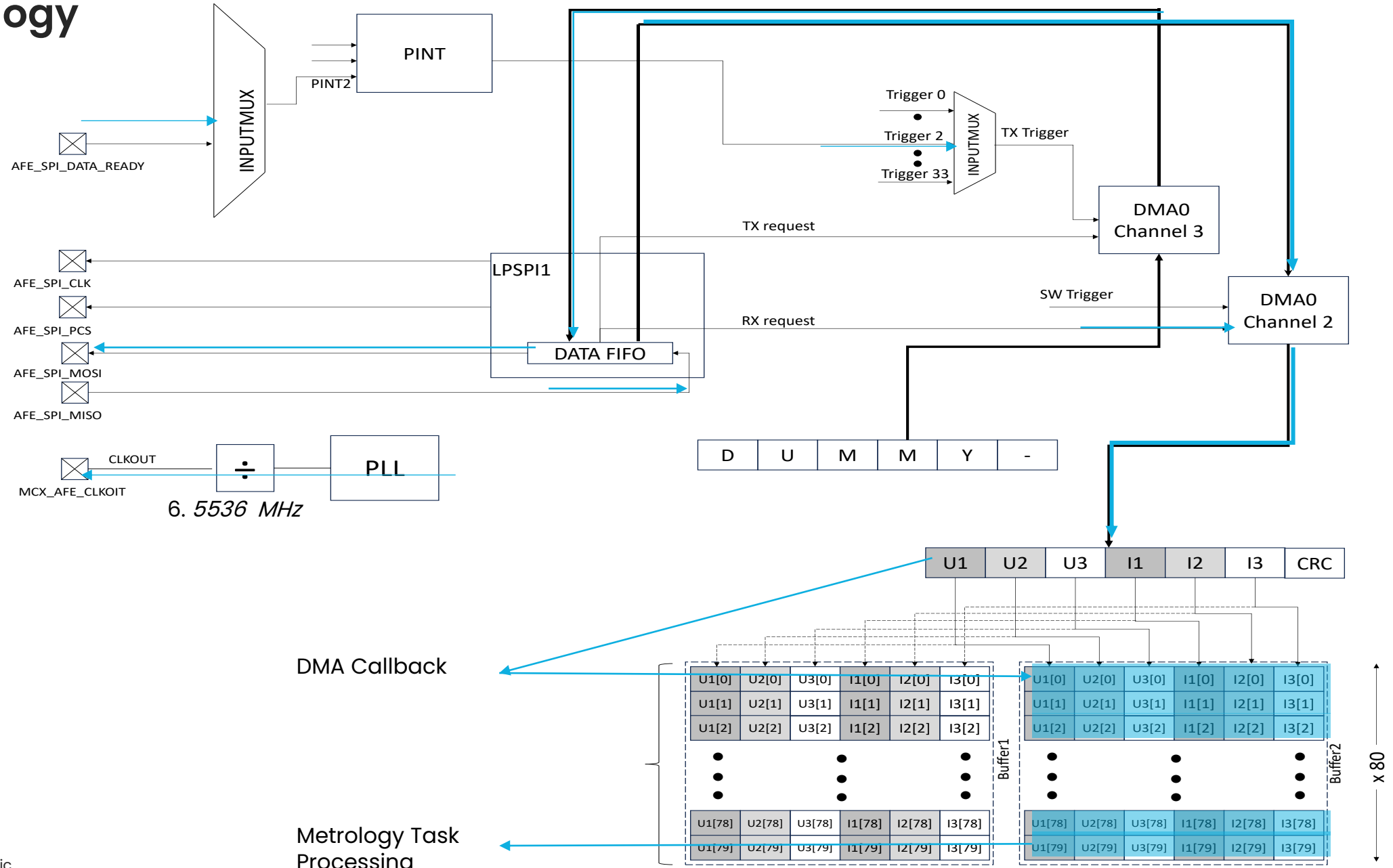
- It must not be connected directly to the power grid

ℹ️ The RW612 runs the metrology over simulated data, but the calculation process is real



NXP Public

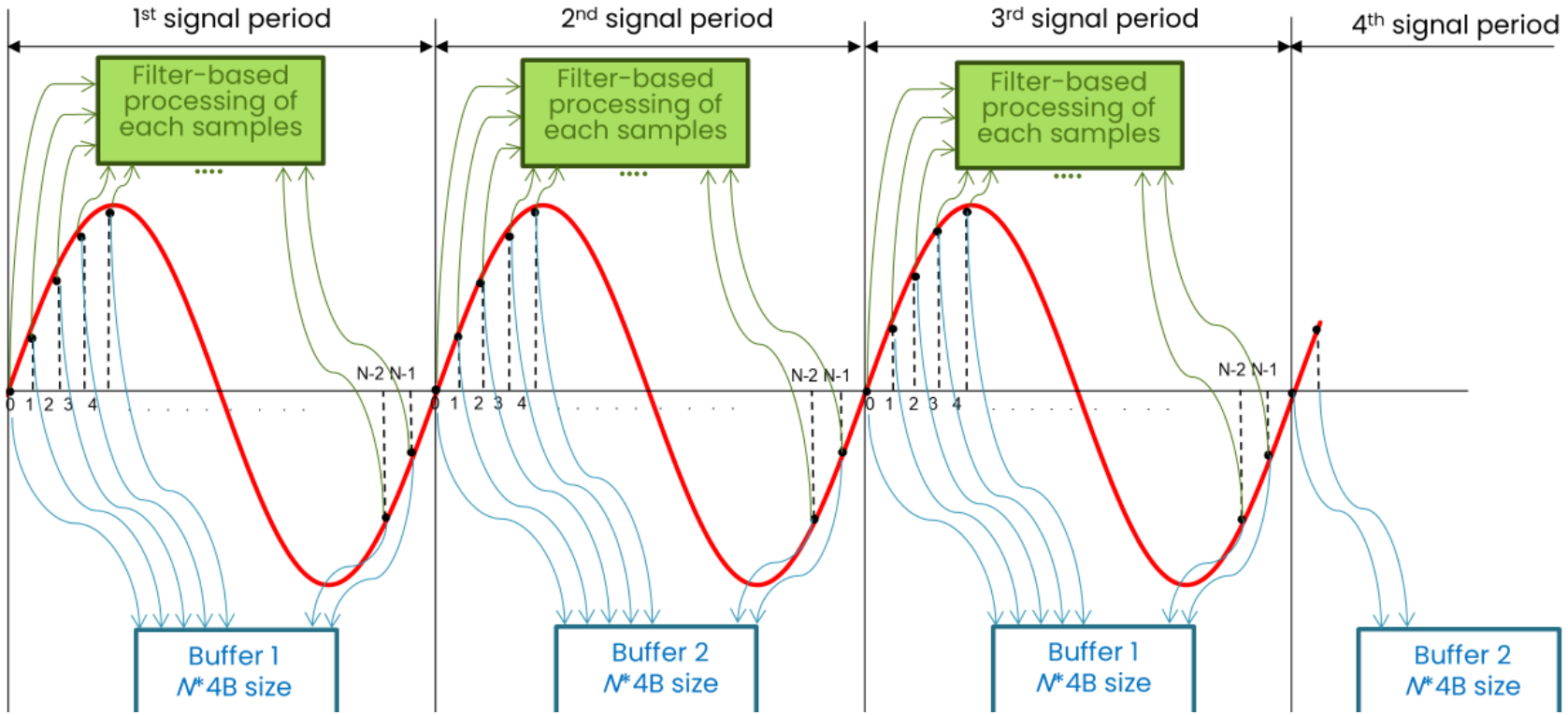
Metrology



NXP Public

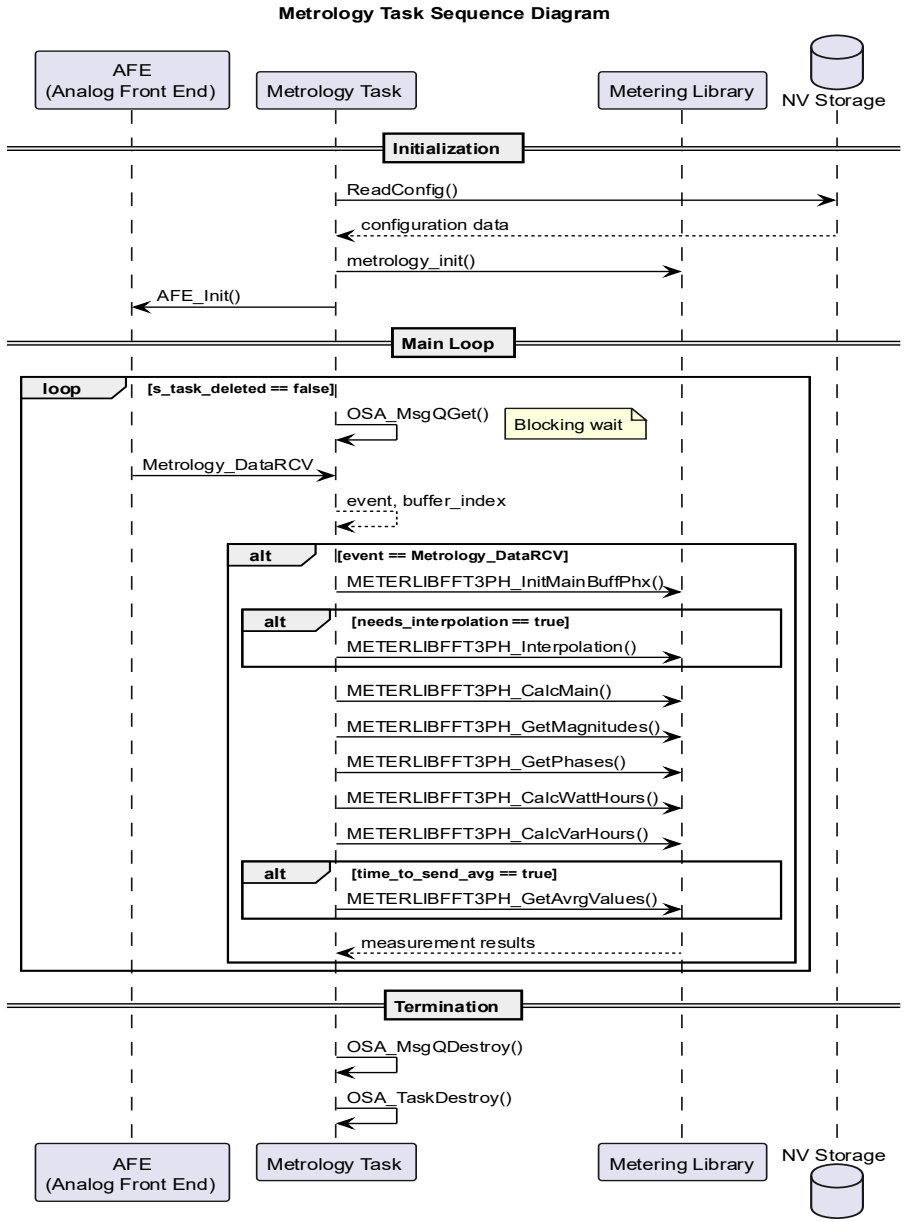
Metrology

SAMPLING & FFT-based data processing



NXP Public

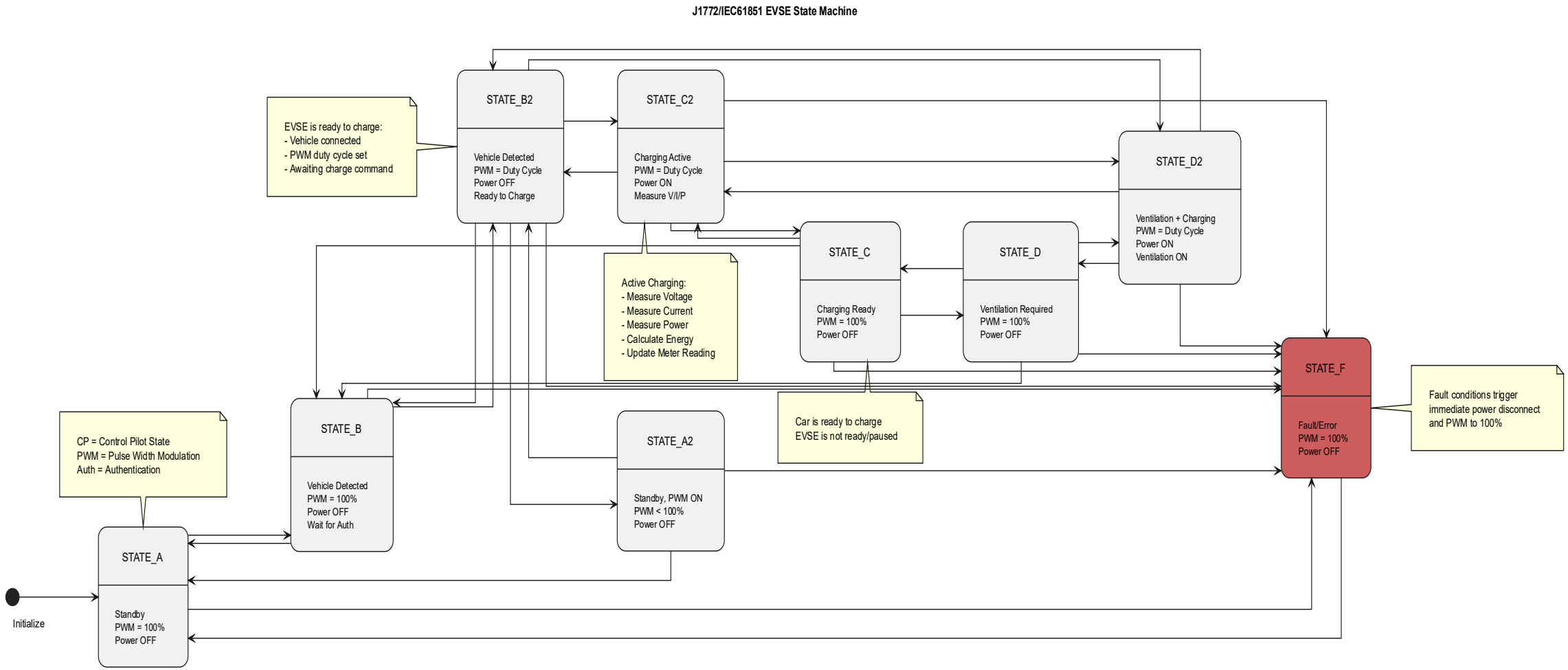
Metrology



API	Description
EVSE_MetrologyInit()	It initializes and starts the Metrology Task
EVSE_MetrologyDeinit()	Requests graceful shutdown of the Metrology Task
Metrology_ReadCurrent()	Returns current RMS values for all phases
Metrology_ReadVoltage()	Returns voltage RMS values for all phases
Metrology_ReadActivePower()	Returns accumulated energy values
Metrology_ReadEnergy()	Returns accumulated energy values
Metering_SampleReset()	Reset the metering sample buffer.
Metering_SampleAdd()	Add voltage and current samples for three phases to the metering buffer.
Metering_StartProcessing()	Starts the metering data processing. This will trigger the metrology process to run.
Metering_SampleFinish()	Finalizes the metering sample collection. This will lock the buffer and prepare it for processing.

NXP Public

J1772



NXP Public

06

Open Charge Point Protocol (OCPP) communication



Open Charge Point Protocol (OCPP) - connecting EVs to the grid

OCPP is a standardized communication protocol that enables charging stations to talk with management systems. The protocol was created by the Open Charge Alliance (OCA), an industry consortium dedicated to promoting open standards in EV charging.

- OCPP brings **interoperability** to the charging ecosystem, allowing different vendors' equipment to work together seamlessly and preventing vendor lock-in.
- Any OCPP-compliant charger can communicate with any OCPP-compliant management system, regardless of who manufactured them.

OCPP versions:

- **OCPP 1.5**: Legacy (SOAP)
- **OCPP 1.6**: Most widely deployed (JSON over WebSocket)
- **OCPP 2.0/2.0.1**: Latest standard (more features, ISO 15118 support)

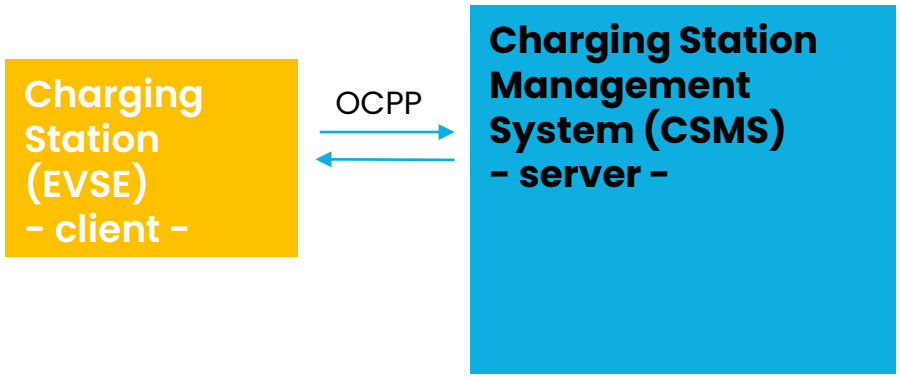


NXP Public

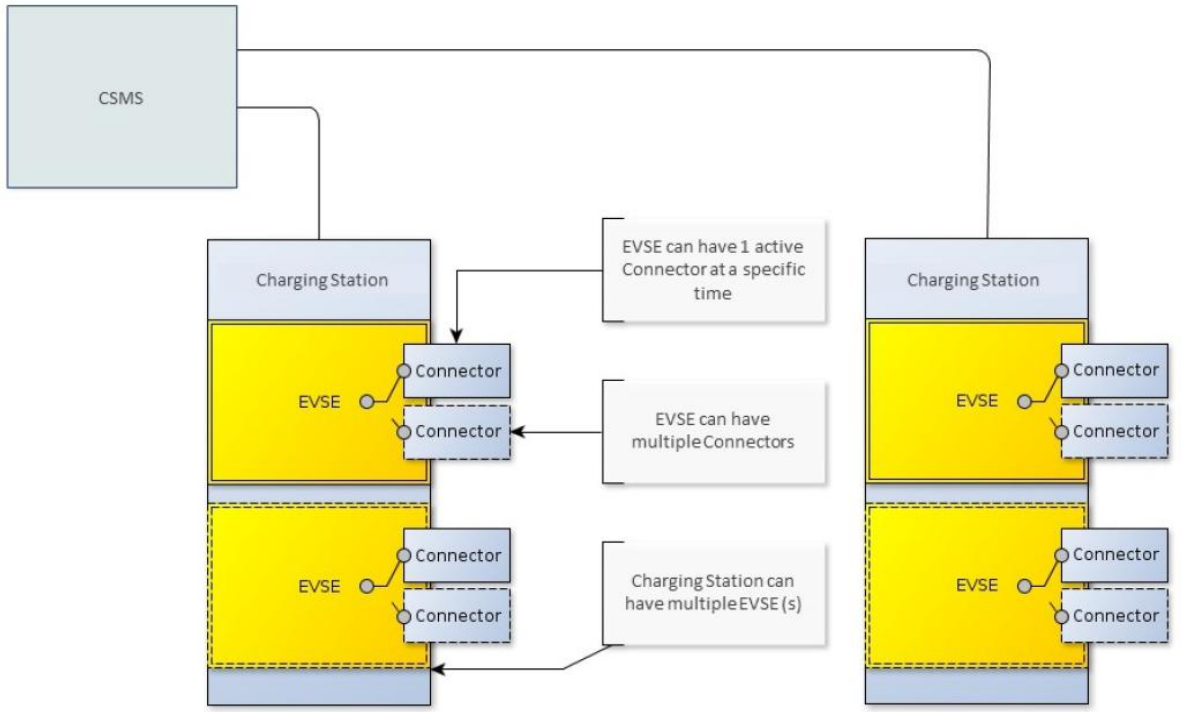
Open Charge Point Protocol (OCPP) - architecture overview

OCPP Architecture Overview

- Charge Point (CP): The physical charging station
- Central System (CS) / CSMS: Backend management system
- Communication flow: Bidirectional (bidirectional communication happens over a persistent connection, typically using WebSocket protocol, which keeps the channel open for real-time messaging in both directions)



Basic OCPP topology



3-tier model as used in OCPP

NXP Public

Open Charge Point Protocol (OCPP) - use cases (I)

OCPP is the protocol used between a **Charging Station** and a **Charging Station Management System** in an EV charging infrastructure in the form of use cases.

The OCPP client implementation relies on the libocpp implementation from EVerest which implements the full set of use cases for OCPP 2.0.1.

Supported OCPP use cases in RW612 application:

- B. Provisioning:
  - B01 - Cold Boot Charging (connect with the CSMS and configure the heartbeat message)
- C. Authorization:
  - C01 - EV Driver Authorization using RFID
- E. Transactions:
  - E01 - Start Transaction options
  - E02 - Start Transaction - Cable Plugin First
  - E06 - Stop Transaction options
  - E09 - When cable disconnected on EV-side: Stop Transaction
  - E10 - When cable disconnected on EV-side: Suspend Transaction
  - E11 - Connection Loss During Transaction

NXP Public

Open Charge Point Protocol (OCPP) - use cases (II)

Supported OCPP use cases:

- E. Transactions:
  - E13 - Transaction-related message not accepted by CSMS
  - E14 - Check transaction status
  - E15 - End of charging process
  - E16 - Transactions with fixed cost, energy, SoC or time
- F. Remote Control:
  - F01 - Remote Start Transaction - Cable Plugin First
  - F02 - Remote Start Transaction - Remote Start first
  - F03 - Remote Stop Transaction
- G. Availability (Charging Station to inform CSMS of its availability for starting new transaction):
  - G01 – Report connector AvailabilityState
  - G02 – Heartbeat
  - G03 - Change Availability EVSE/Connector
- I. Tariff and Cost:
  - I02 - Show EV Driver Running Total Cost During Charging
  - I03 - Show EV Driver Final Total Cost After Charging
- J. Meter Values:
  - J02 - Sending transaction related Meter Values
  - J03 - Charging Loop with metering information exchange

NXP Public

Open Charge Point Protocol (OCPP) - use cases (III)

Not enabled OCPP use cases in the RW612 phase:

- D – Local Authorization List Management
- H - Reservation
- K – Smart charging
- L - Firmware management
- M – Certificate management

The OCPP client implementation addresses these use cases, but they are not enabled in the RW612 application.

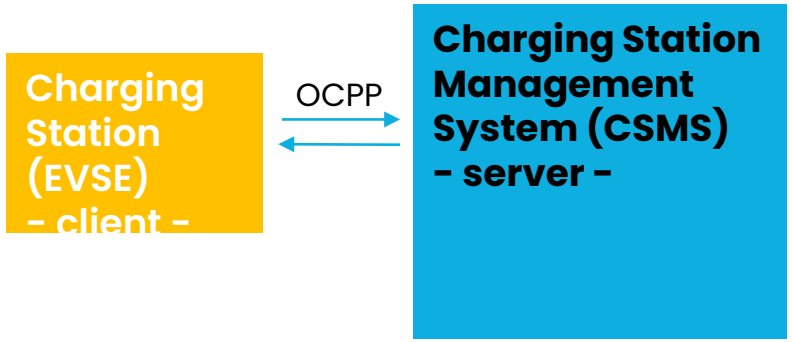
NXP Public

OCPP client implementation (I) - protocol and application layers

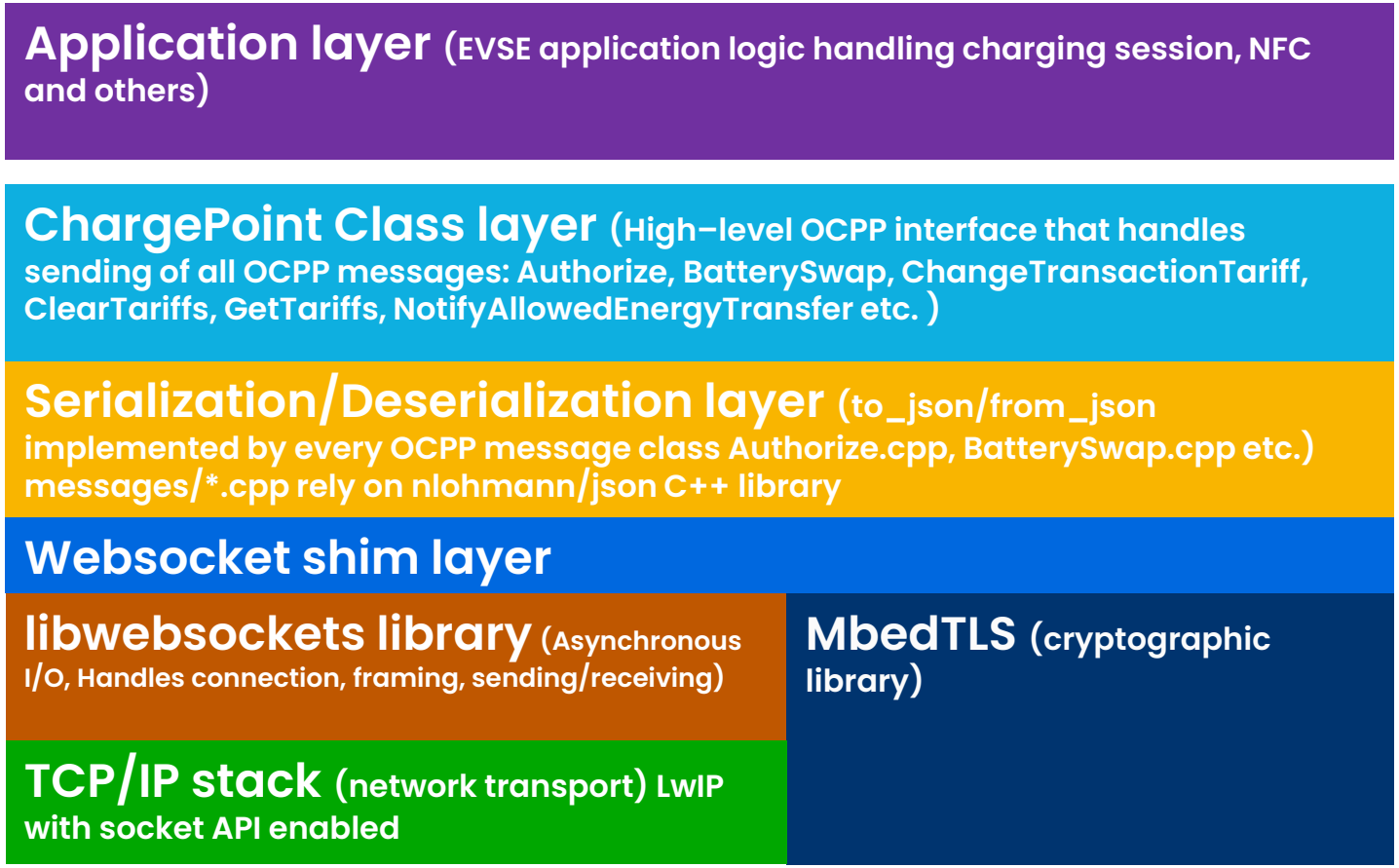
Application layer:
- Handles EV logic and provisioning of the charging point

ChargePoint class layer:
- Handles sending/receiving of all standard OCPP messages

Serialization/Deserialization layer:
- Packs/Unpacks the data in json format



Basic OCPP topology



OCPP stack architecture

NXP Public

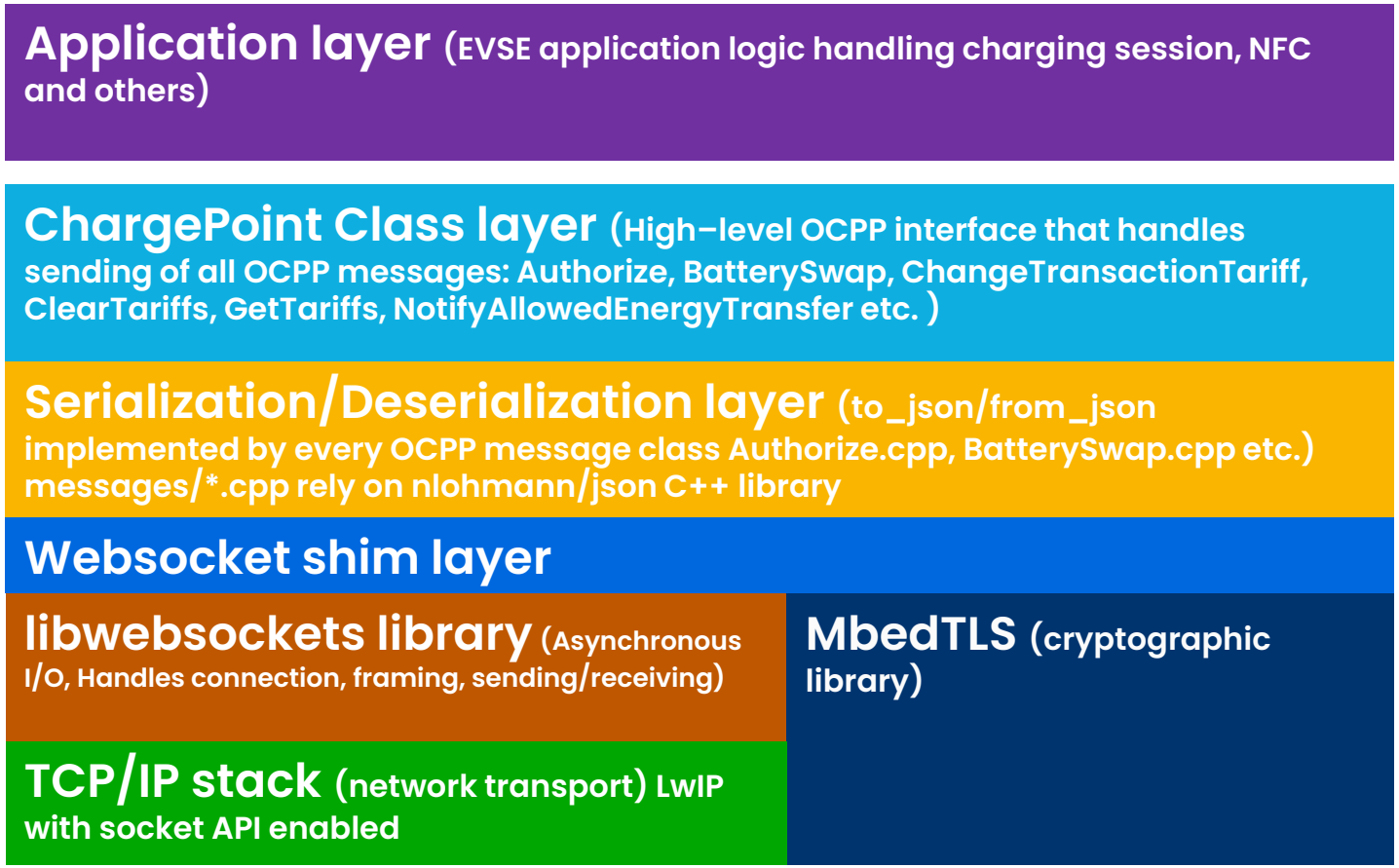
OCPP client implementation - underlying connection (I)

Websocket shim layer:

- Handles the initialization of a WebSocket connection and starts OCPP internal threads handling the connection

WebSocket connection:

- Persistent connection (not request/response like HTTP)
- Charger initiates connection to CSMS
- Benefits: Real-time, bidirectional, efficient
- Connection management: Reconnection logic, heartbeats



OCPP stack architecture

NXP Public

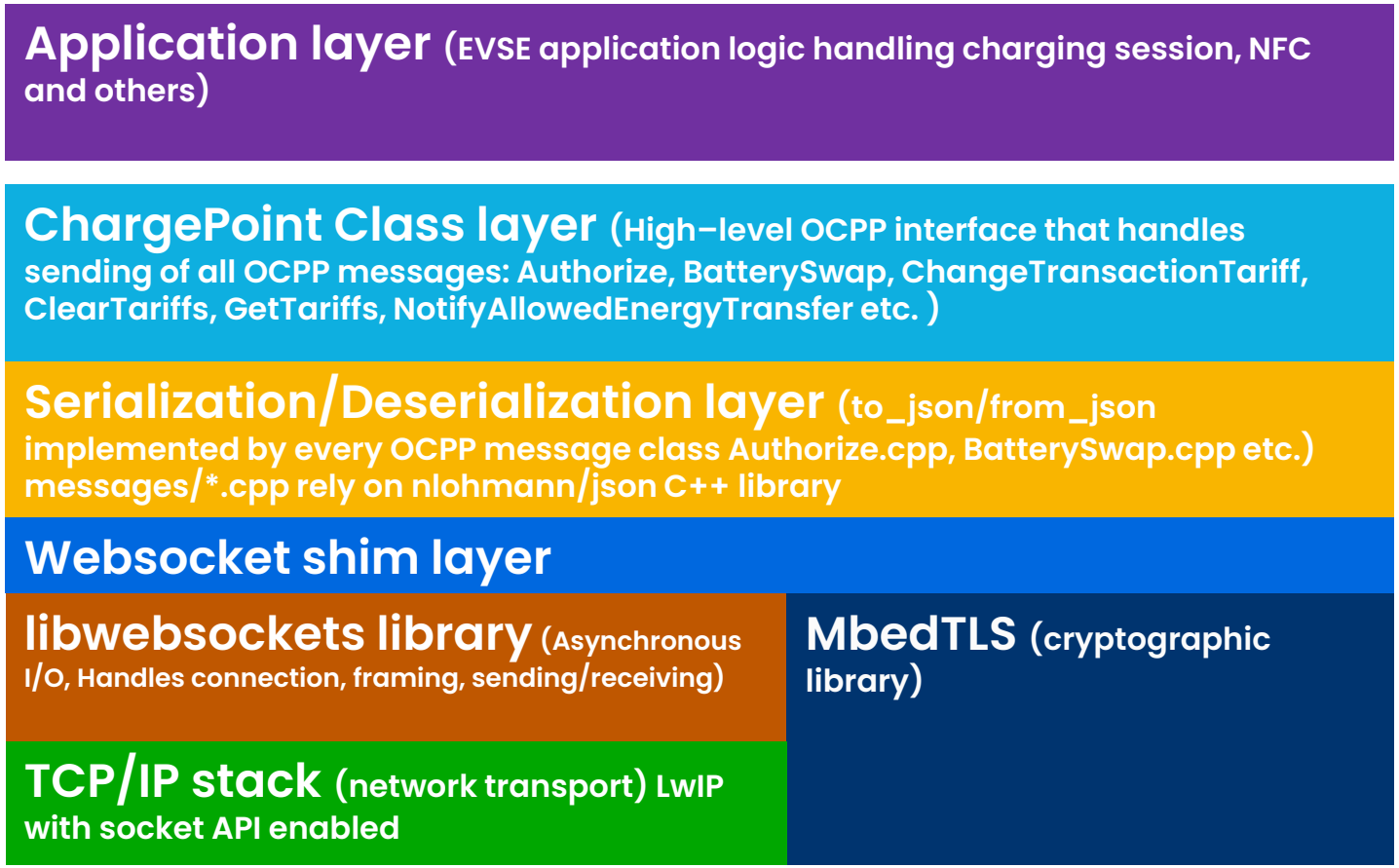
OCPP client implementation - underlying connection (II)

TCP/IP connection

- Implemented by the LwIP stack with socket API enabled

MbedTLS

- Used for cryptographic operations needed for TLS in OCPP Security Profiles 2 and 3.
- Includes SE050 support for the OCPP client's private key.

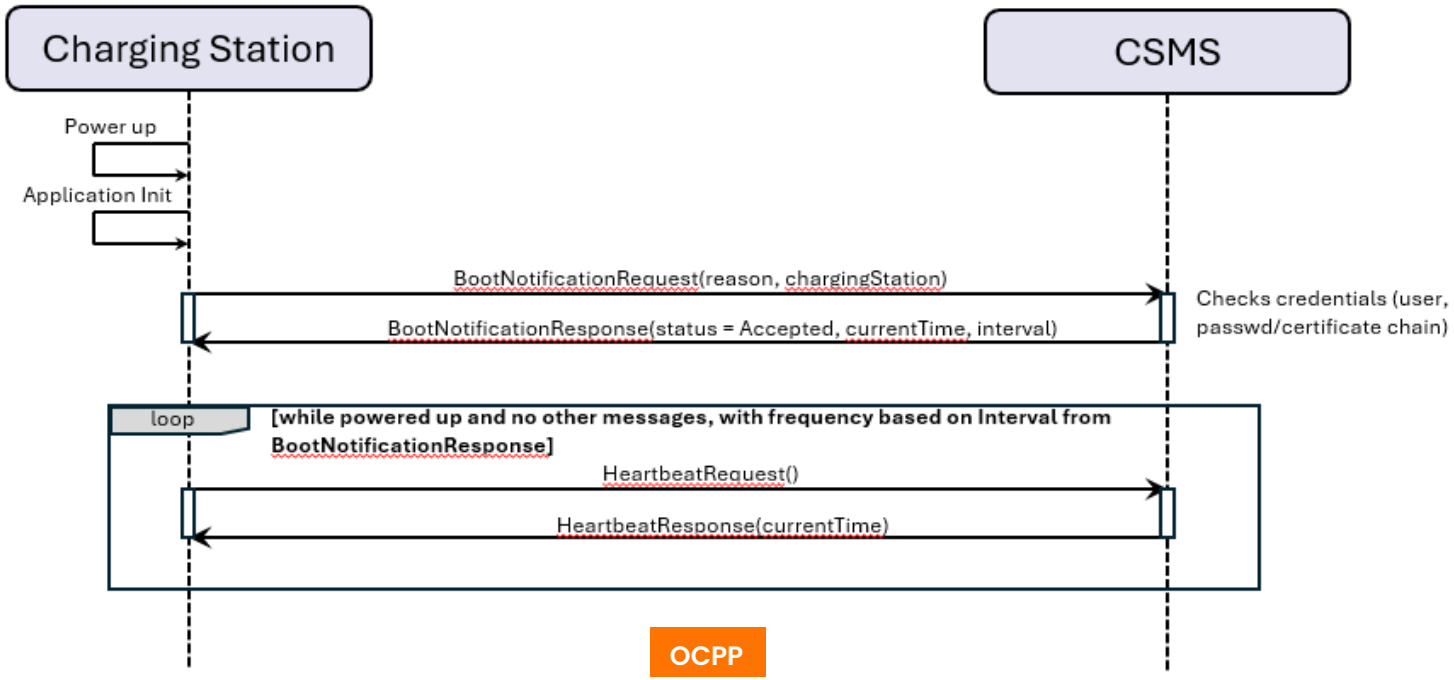


OCPP stack architecture

NXP Public

Open Charge Point Protocol (OCPP) - communication flow

Real-world communication flow:
Charger boots up → BootNotification request
CSMS accepts → BootNotification response

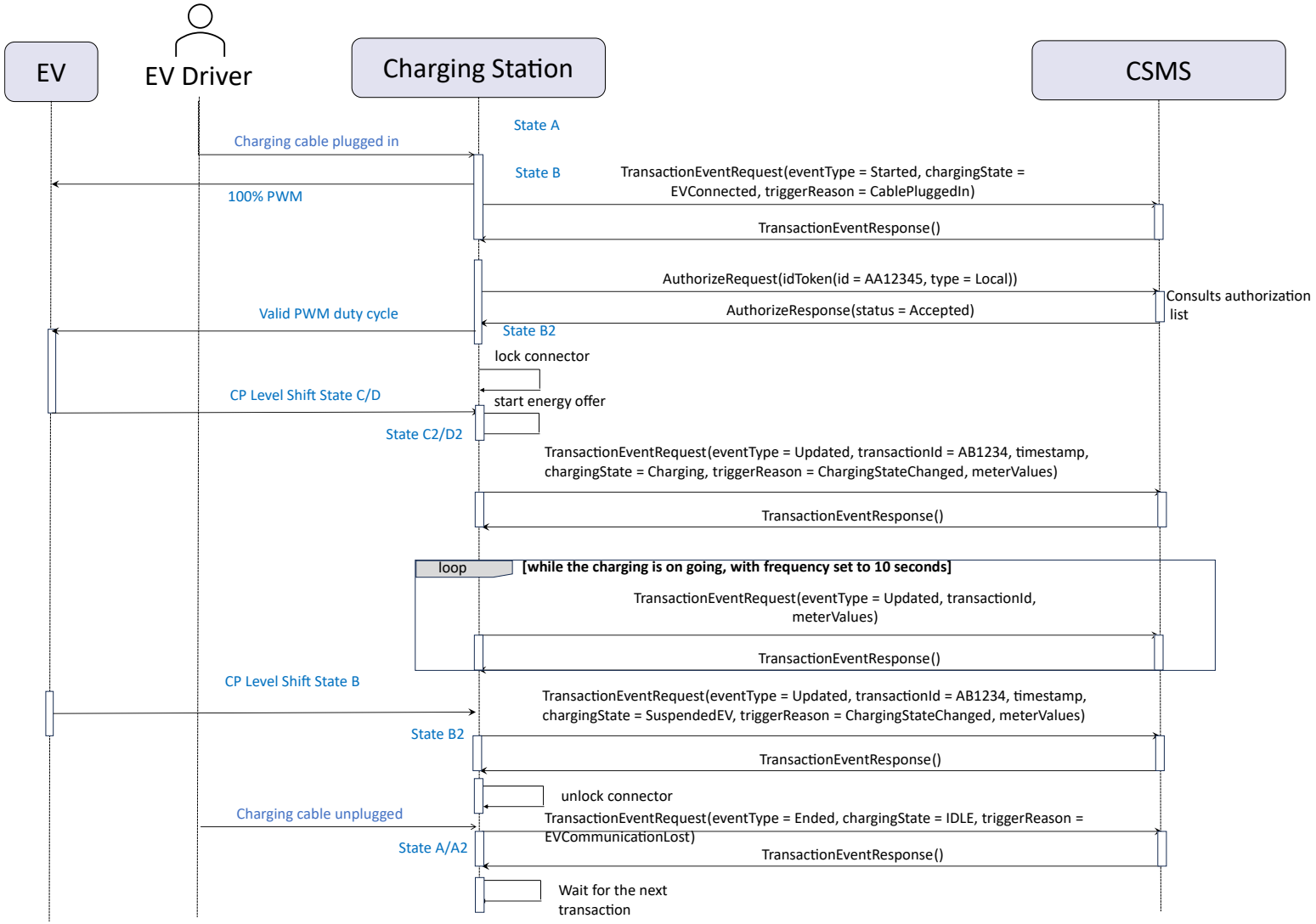


Provisioning the CS, handling the OCPP connection

NXP Public

Open Charge Point Protocol (OCPP) - communication flow

State	OCPP EVENT
State A	CLEAN SESION
State A2	EVCommunicationLost. IDLE
State B	CablePluggedIn EVConnected
State B2	ChargingStateChanged SuspendedEV
State C/D	ChargingStateChanged SuspendedEVSE
State C2/D2	ChargingStateChanged Charging



NXP Public

Open Charge Point Protocol (OCPP) - security profiles

Security profiles in OCPP define the level of security and authentication used for communication between the charging station and the CSMS. They establish how the connection is secured and what authentication mechanisms are required.

Security Profile 1: Unsecured HTTP with Basic Authentication
Security Profile 2: Secured WebSocket with Basic Authentication
Security Profile 3: Secured WebSocket with Client Certificates (TLS 1.2 mutual authentication)

OCPP 2.0.1: All production deployments must use encrypted connections (Security Profile 2 or 3).

How Security Profiles Work in Practice
When a charging station boots up, it's configured with a specific security profile. This configuration determines:

- **Connection URL**: Whether to use ws:// (unsecured) or wss:// (secured)
- **Authentication method**: Basic auth credentials or client certificates
- **Certificate validation**: Which CA certificates to trust
- **TLS settings**: Cipher suites, protocol versions, etc.

The CSMS is also configured to expect connections at a certain security level. If a charging station tries to connect with a lower security profile than the CSMS requires, the connection will be rejected.

NXP Public

Open Charge Point Protocol (OCPP) - CSMS setup

CitrineOS is an open-source Charging Station Management System (CSMS) designed to manage electric vehicle (EV) charging networks. It implements the Open Charge Point Protocol (OCPP) 2.0.1, which facilitates communication between charging stations and charge management software.

UG: [UG10362: Charging Station Management System (CSMS) Installation and Configuration | NXP Semiconductors]()

Build the docker images for the CitrineOS core and CitrineOS UI on your Unix system:
sudo docker compose up –build –d

```
a0ae4f@~$ sudo docker ps -a
CONTAINER ID   IMAGE                                          COMMAND                  CREATED         STATUS                    PORTS
                                                                                                                                                                                    NAMES
0aa0ae4fd11f   ghcr.io/home-assistant/home-assistant:stable   "/init"                  2 months ago    Up 2 weeks
                                                                                                                                                                                    homeassistant
684765712f7a   server_citrine                                 "docker-entrypoint.s…"   3 months ago    Up 2 weeks                0.0.0.0:8080-8082->8080-8082/tcp, [::]:8080-8082->8080-8082/tcp, 0.0.0.0:8085->
8085/tcp, [::]:8085->8085/tcp, 0.0.0.0:8092->8092/tcp, [::]:8092->8092/tcp, 0.0.0.0:8443-8444->8443-8444/tcp, [::]:8443-8444->8443-8444/tcp, 0.0.0.0:9229->9229/tcp, [::]:9229->9229/tcp   server_citrine_1
41d4d24c6eff   minio/mc                                       "/bin/sh -c ' # Enab…"   3 months ago    Exited (0) 2 weeks ago
                                                                                                                                                                                    server_minio-init_1
6c5084b1c520   hasura/graphql-engine:v2.40.3.cli-migrations-v3   "docker-entrypoint.s…"   3 months ago    Up 2 weeks (healthy)      0.0.0.0:8090->8080/tcp, [::]:8090->8080/tcp
                                                                                                                                                                                    server_graphql-engine_
1
ac42f0a401f2   rabbitmq:3-management                          "docker-entrypoint.s…"   3 months ago    Up 2 weeks (healthy)      4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, [::]:5672->5672/tcp, 15671/tcp, 156
91-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, [::]:15672->15672/tcp                                                                                                            server_amqp-broker_1
45da6efcacad   minio/minio                                    "/usr/bin/docker-ent…"   3 months ago    Up 2 weeks (healthy)      0.0.0.0:9000-9001->9000-9001/tcp, [::]:9000-9001->9000-9001/tcp
                                                                                                                                                                                    minio
97d8ecc4add9   postgis/postgis:16-3.5                         "docker-entrypoint.s…"   3 months ago    Up 2 weeks (healthy)      0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp
                                                                                                                                                                                    server_ocpp-db_1
```

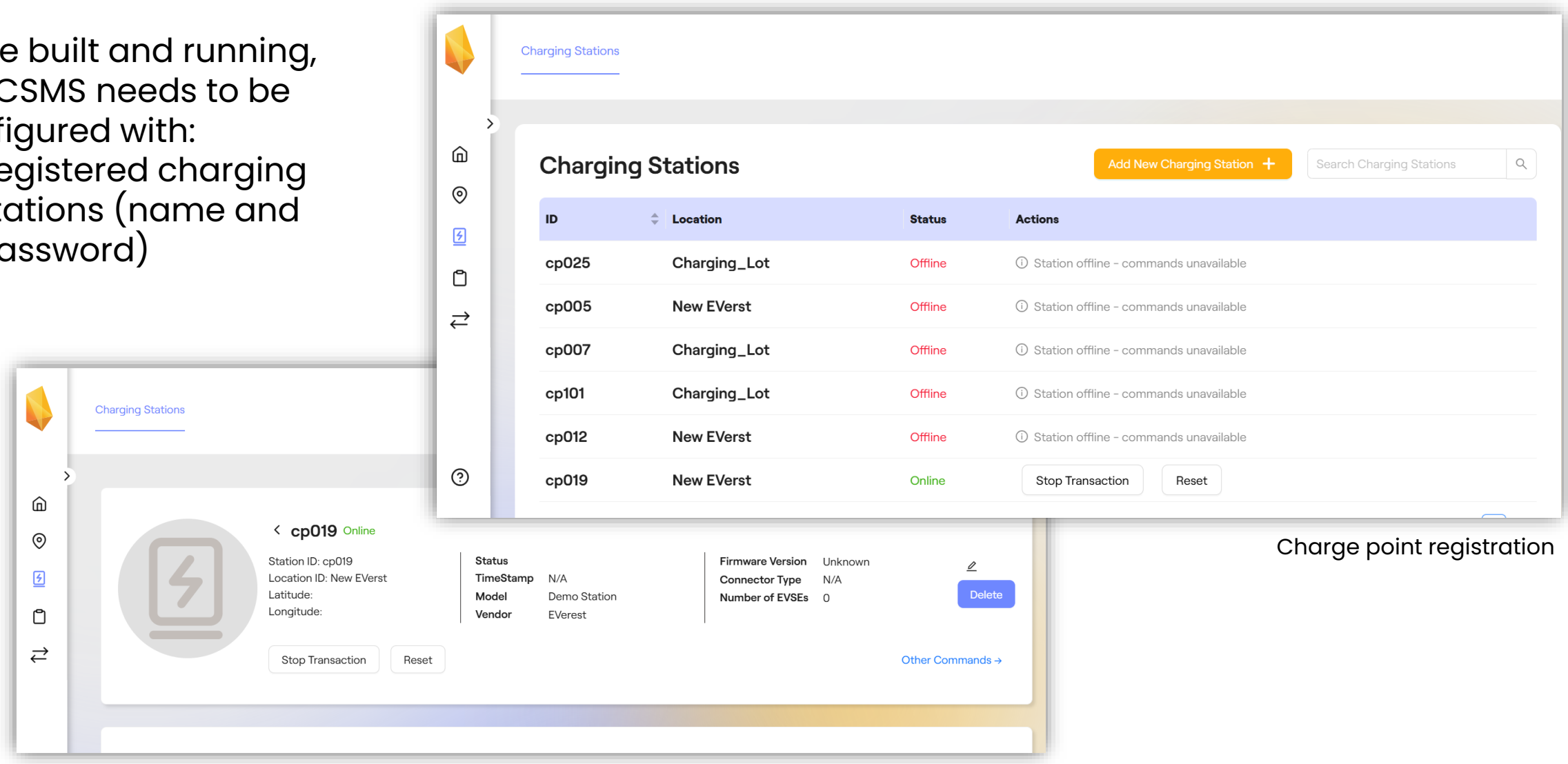
Active docker containers for CitrineOS core

NXP Public

Open Charge Point Protocol (OCPP) - CSMS configuration

Once built and running, the CSMS needs to be configured with:

- Registered charging stations (name and password)



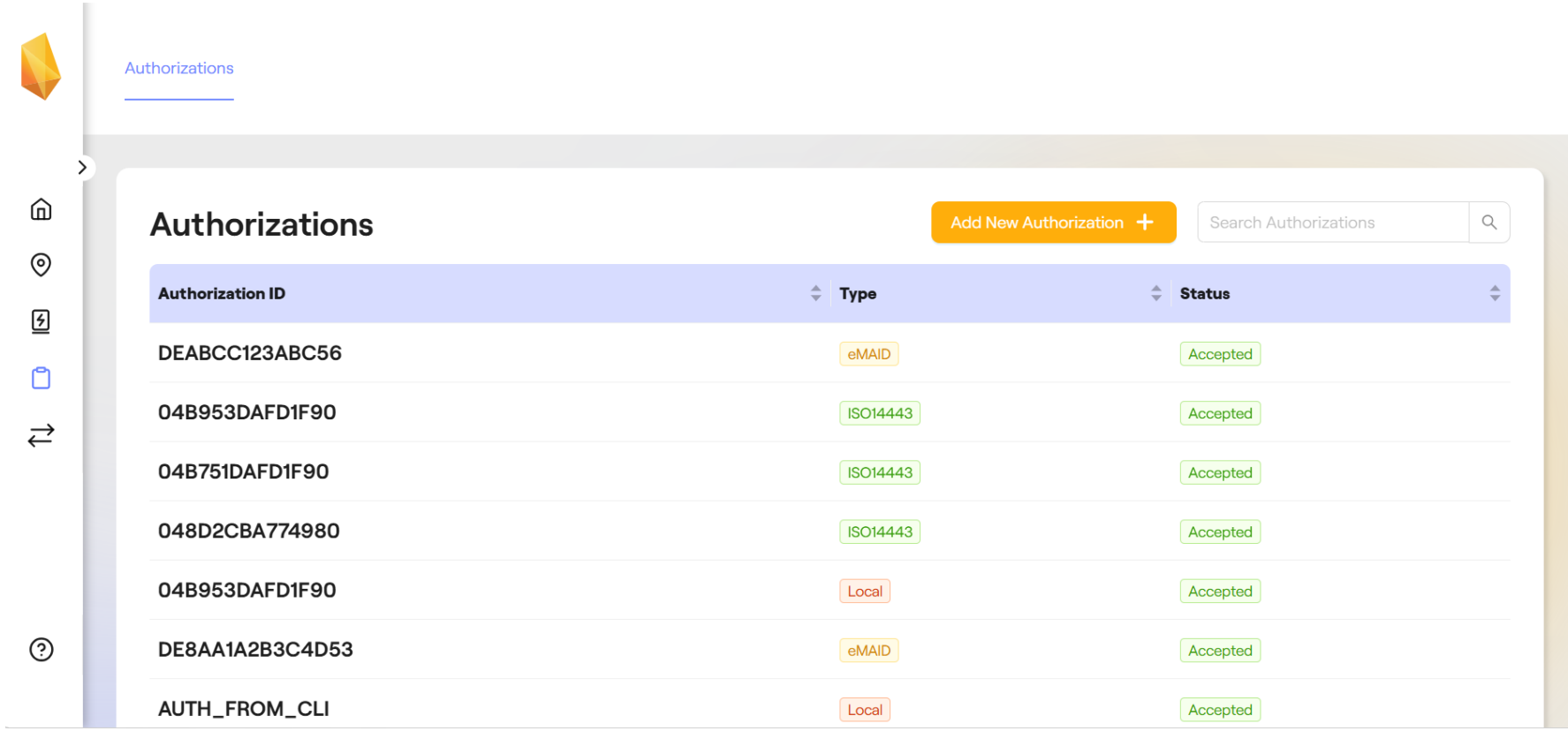
Charge point registration

NXP Public

Open Charge Point Protocol (OCPP) - CSMS configuration

Once built and running, the CSMS needs to be configured with:

- Authorization list (with the accepted local value)

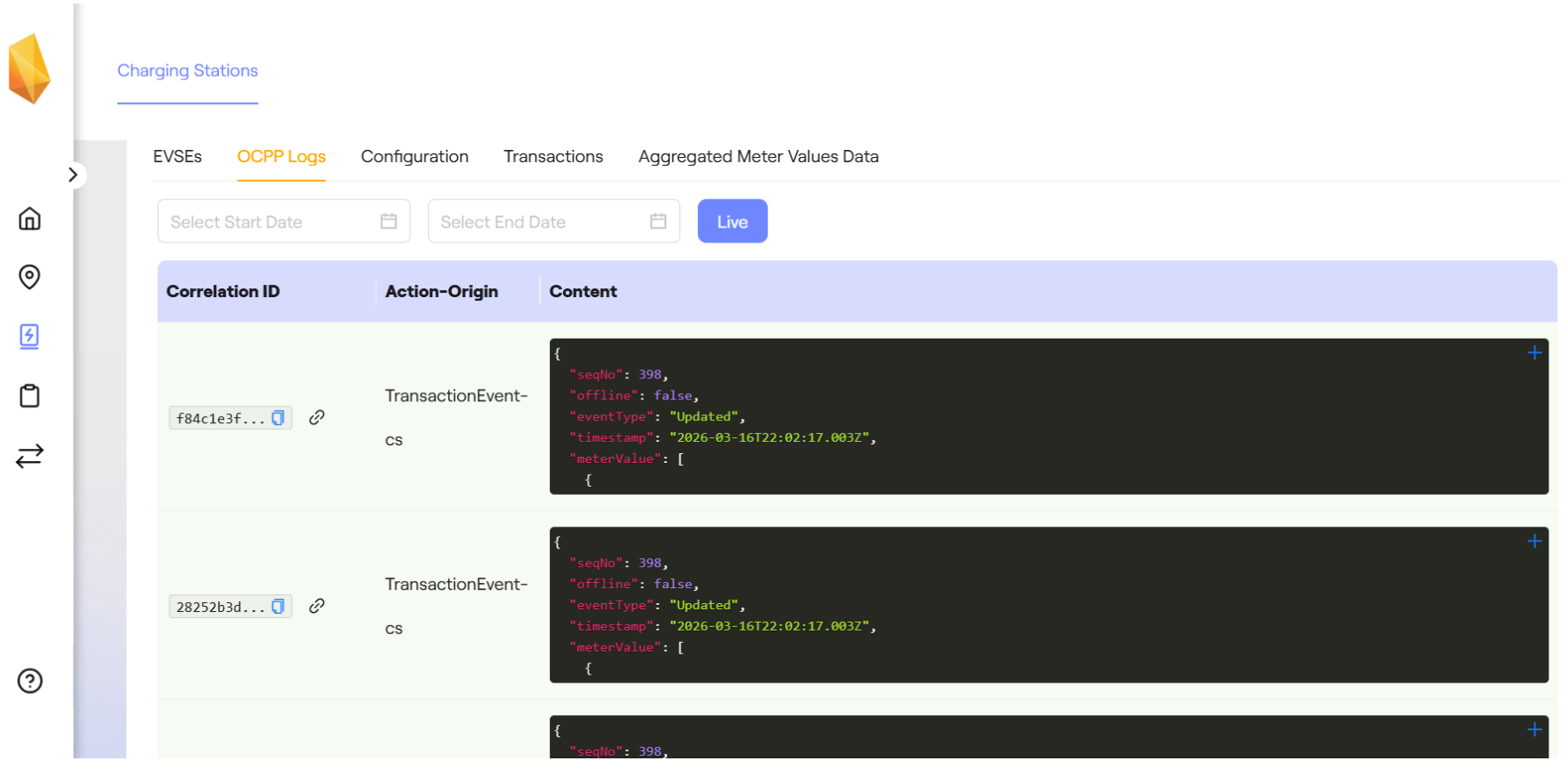


Authorization list configured on the CSMS

NXP Public

Open Charge Point Protocol (OCPP) - CSMS configuration

At CSMS level, the metering values reported by the EVSE are visible in the logs associated with the monitored transaction/charging session.

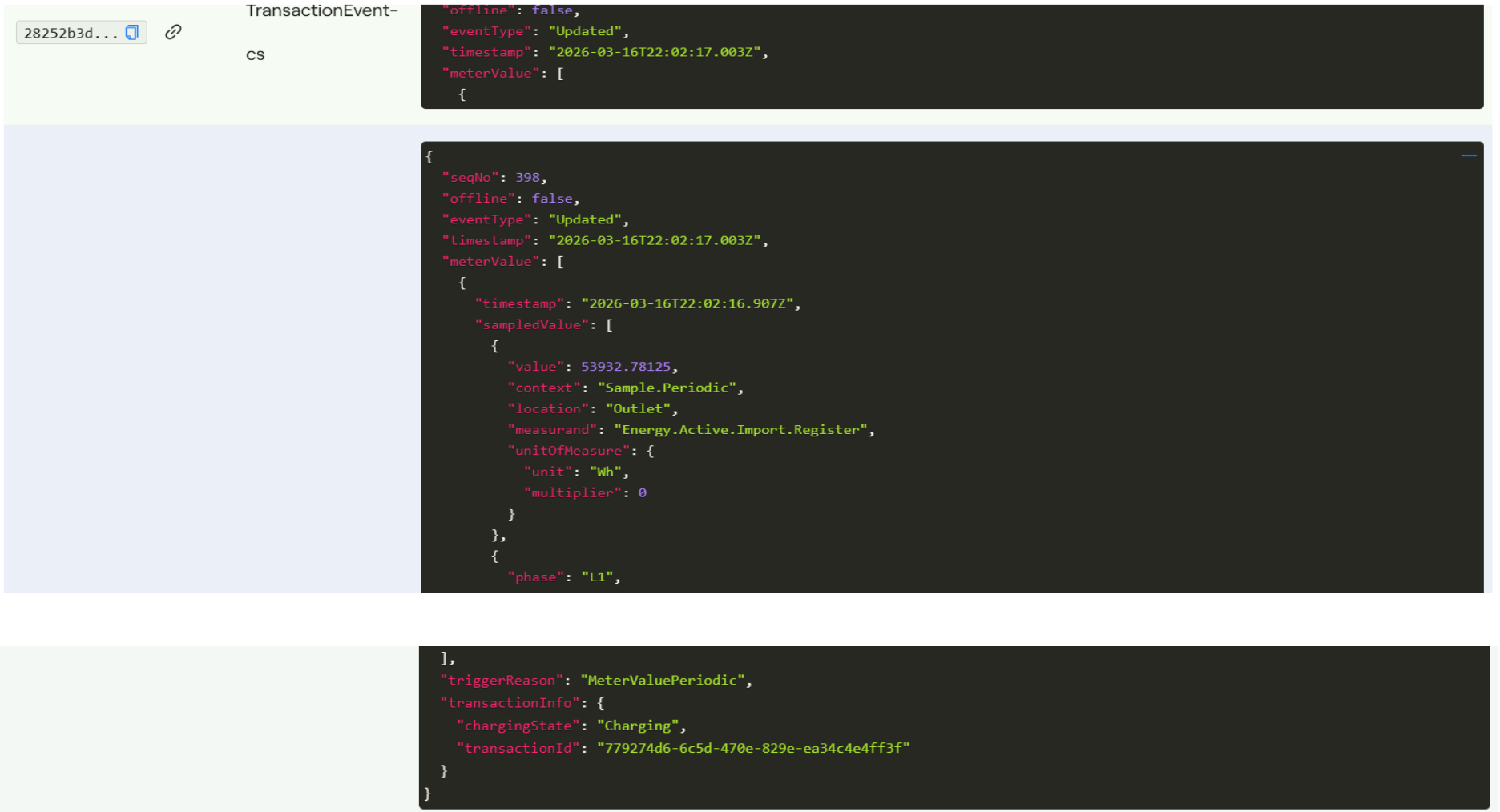


Monitored charging session data in the CSMS logs

NXP Public

Open Charge Point Protocol (OCPP) - CSMS configuration

At CSMS level, the metering values reported by the EVSE are visible in the logs associated with the monitored transaction/charging session.



Monitored charging session data in the CSMS logs

NXP Public

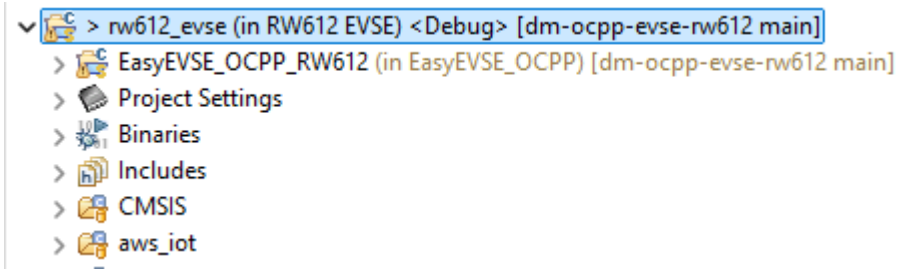
07

OCPP app – build



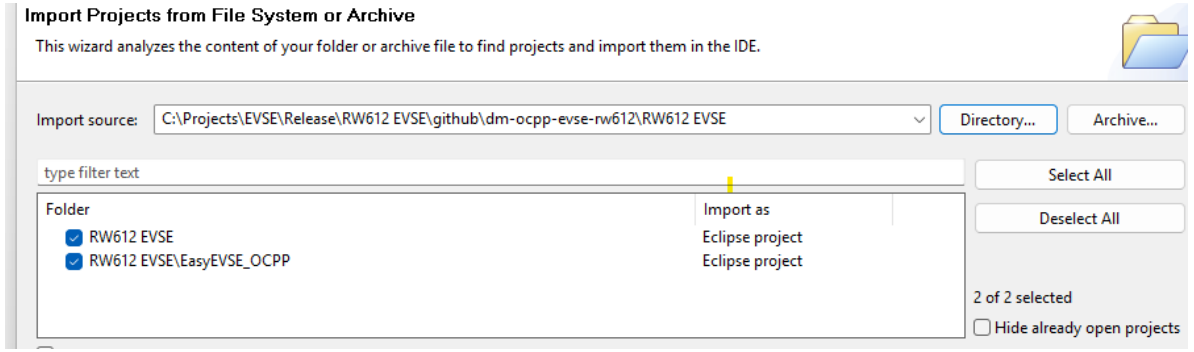
EVSE RW612 build

Import the project in MCUXpresso IDE or VS Code



To import the project in MCUXpresso IDE:

1. Proceed to import the project from filesystem
2. Choose the previously cloned archive
3. Go to project selection
4. MCUXpresso IDE or VS Code will automatically detect 2 projects:
   - OCPP project with the OCPP stack implementation for the EVSE application.
   - EasyEVSE project with Debug build configuration.



NXP Public

Demo

EasyEVSE Entry-level Development Platform (RW612 MCU, FreeRTOS, Wi-Fi 6)

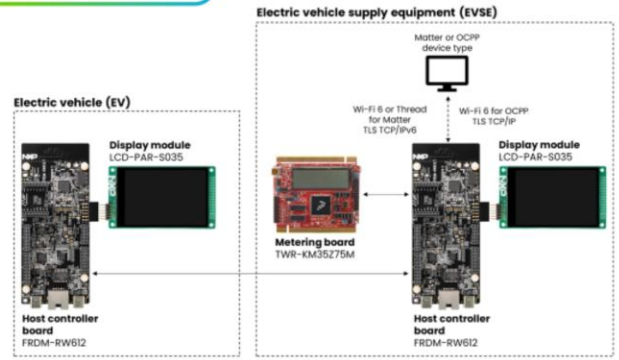
NXP accelerates fast, differentiated entry-level single- and multi-phase AC EV charger development

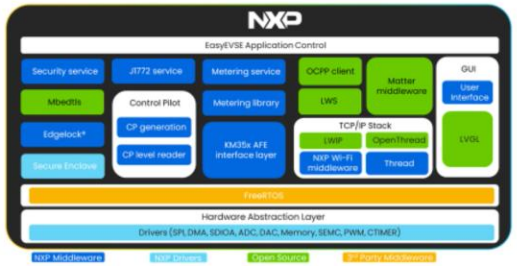
- **Reduce design, complexity, cost** and **footprint** by converging **4** chips into **1**
- Benefit from industry-leading **1x1 dual-band Wi-Fi 6** for **extended range**, **higher data throughput**, and **robust connectivity**
- **Matter-enabled** EV charging applications over **Wi-Fi 6** or **Thread**
- Ensure seamless communication between EVSE and cloud services using the proven **open-source EVeret OCPP 2.0.1** implementation
- Enable **proven resilient EdgeLock Secure Enclave** for secure backend communications
- Achieve accurate, reliable billing and advanced power-line monitoring using **modifiable, pre-certified metrology** software
- Enable **application software reuse** to scale up to higher-performance EasyEVSE MCU and MPU platforms

10+ 15+
Product Longevity

NXP | Restricted

Solution info: page coming soon





Get started
Application note and Get Started instructions

Development hardware
Order from nxp.com

Complementary development software
Download from github.com/NXP

Solution order numbers	
FRDM-RW612*	Host Controller Board
TWR-KM35Z75M	Metering Board
LCD-PAR-S035*	3.5" 480x320 IPS TFT LCD Module

*Choose one for the EV and one for the EVSE side

NXP components	
RW612	RW612 Wi-Fi/BLE MCU
MKM35Z75M	KM35 metrology MCU

NXP Public

08

Matter



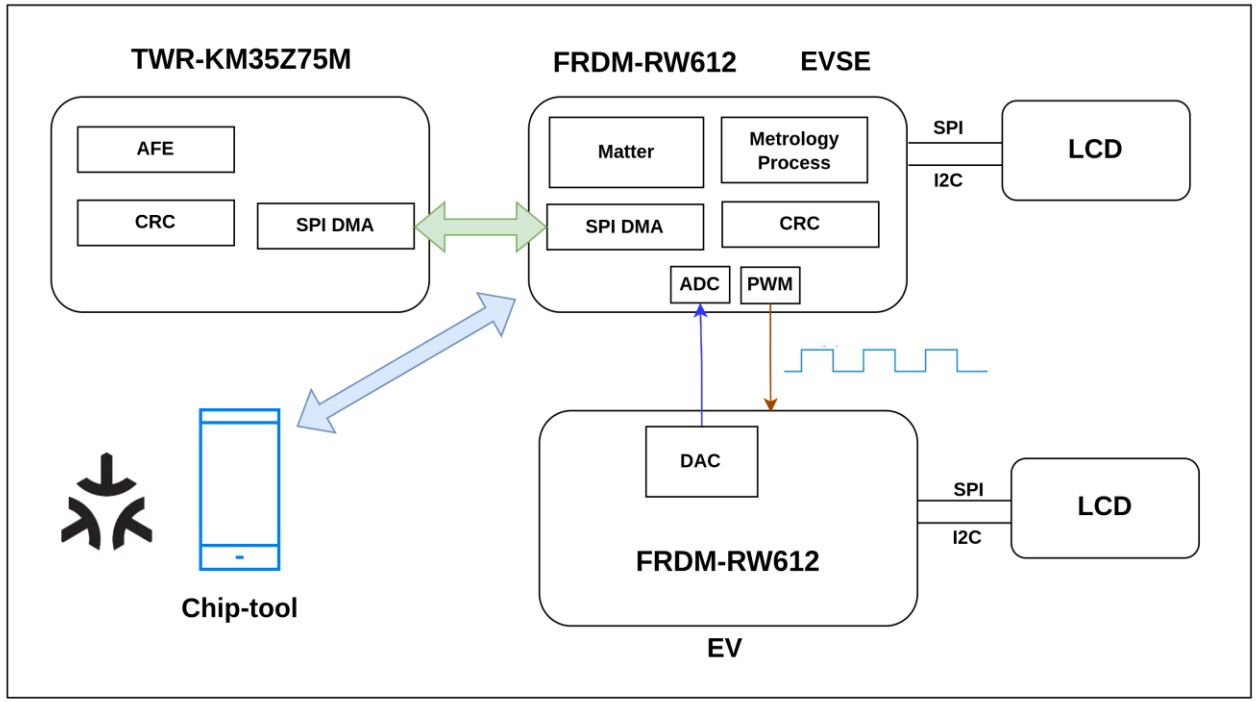
EasyEVSE system architecture : Matter RW612 - PoC

- Matter Controller:
  - Android NXP app/ CHIPtool
  - Matter commands for Charging Start, Stop commands
  - Matter controller can retrieve the EV state, session ID and charging metrics

- EVSE RW612:
  - Matter over Wi-Fi / Thread
  - SPI interface with KM35x, which provides metrology data
  - Generate PWM output as per the EV state (i.e., each state reflects to different Voltage input)
  - LCD to provide UI for Matter Commissioning, reflect the EVSE states

- KM35x:
  - Raw voltage and current data is share using SPI DMA with CRC to RW612 EVSE

- EV RW612 (CAR):
  - LCD to provide UI for simulating car states
  - Based on the car state, generate different Voltage level

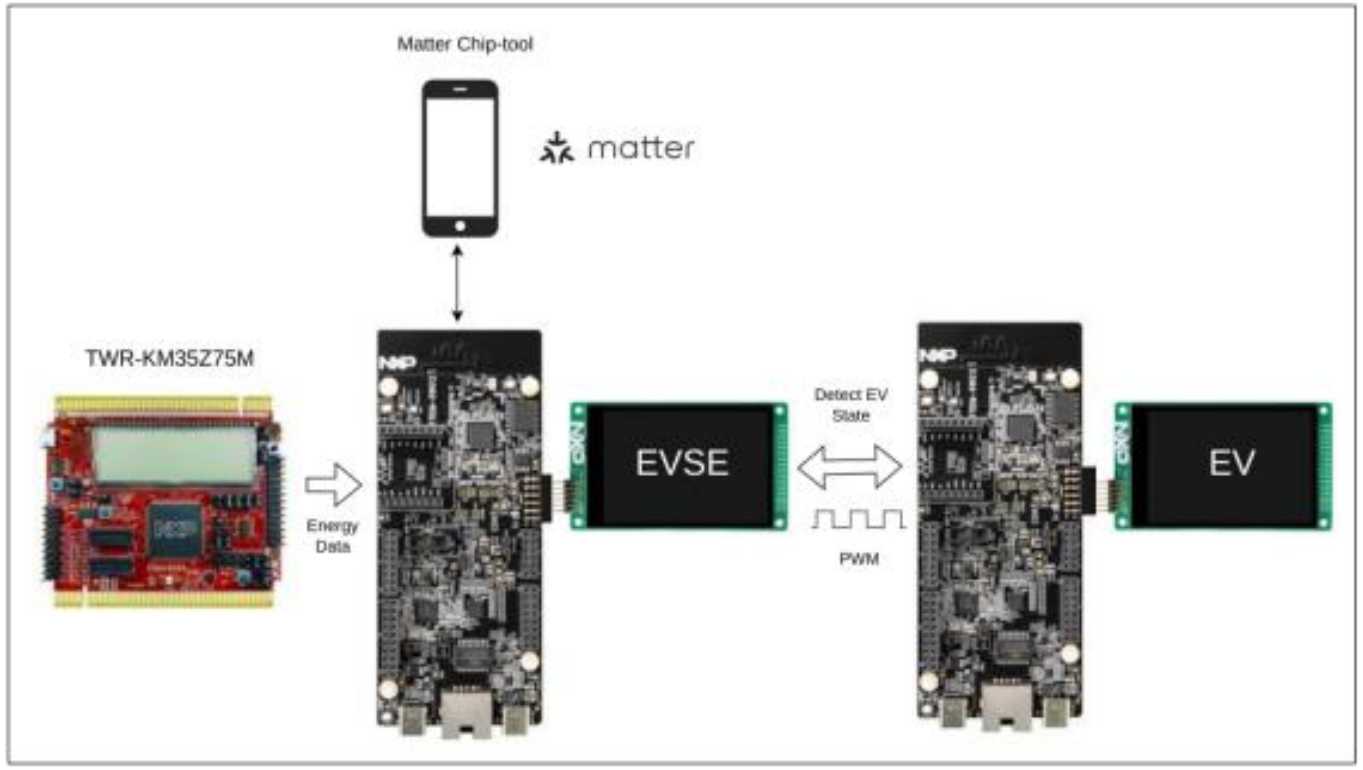


system architecture

NXP Public

EasyEVSE Software Architecuture: Matter RW612 - PoC

- EVSE – EV: The FRDM-RW61x EV sends the different state [i.e. state A, state B, state C, state E, state F] to FRDM-RW61x EVSE and FRDM-RW61x EVSE detects the different voltage level and generates PWM.
- EVSE - CHIPtool: The FRDM-RW61x EVSE commissioned with Matter chip-tool and after commission successfully, Matter CHIPtool sends a charging start/stop/auto schedule command to the FRDM-RW61x EVSE based on state. When FRDM-RW61x EVSE detects fault condition i.e. state F, it updates the fault status to the chip-tool.
- EVSE – KM35x: During the FRDM-RW61x EVSE charging in progress KM35 metering board sends the raw voltage and current data using SPI DMA with CRC check and FRDM-RW61x EVSE receives SPI DMA data then check CRC, process the data and send to the Matter CHIPtool.



NXP Public

09

Matter EVSE cluster



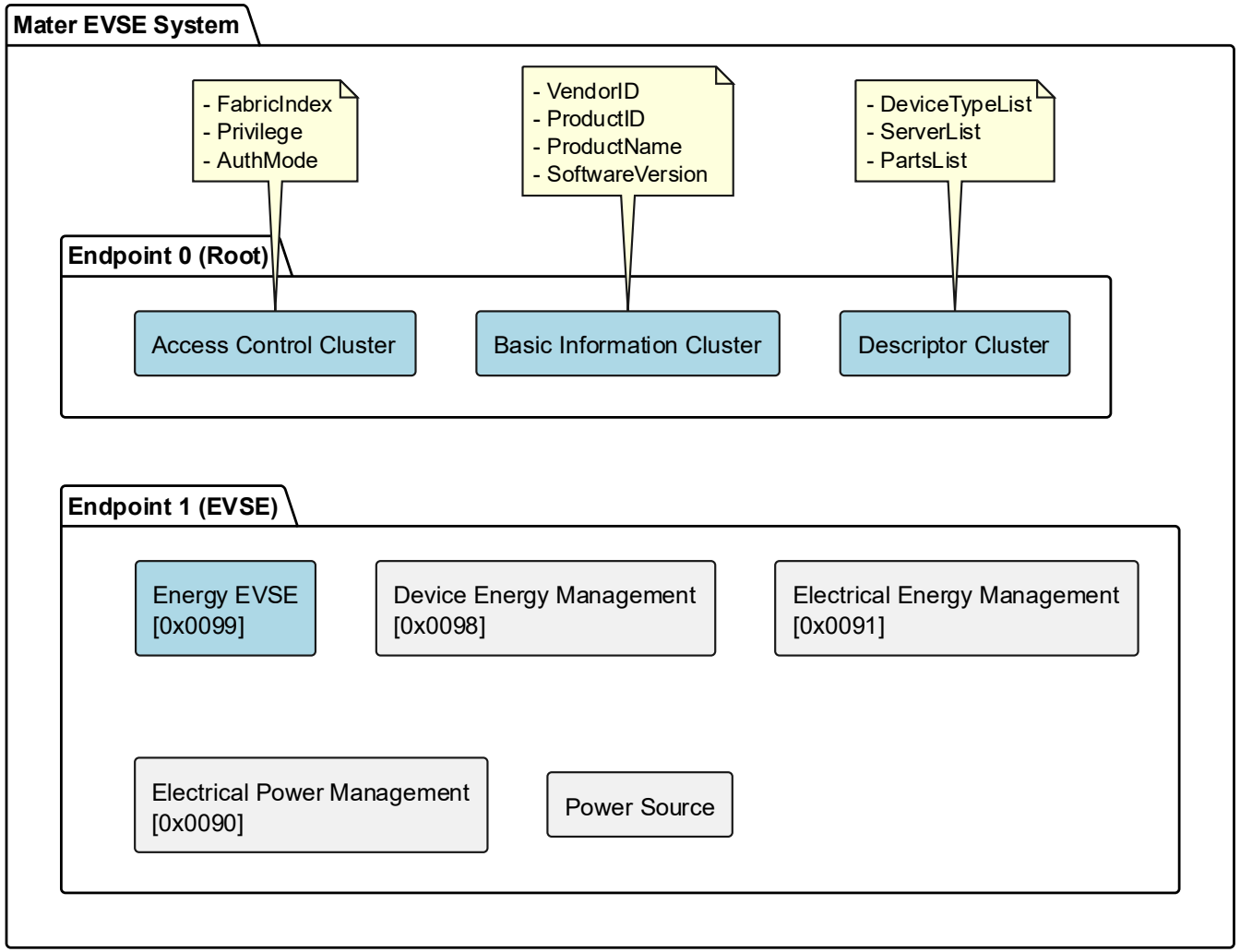
Matter EVSE device type: end points and clusters details

- All Matter devices enables the Root Node, which is required
  - Device capability discovery
  - On-boarding

- RW612 Matter EVSE device PoC enables the Energy EVSE cluster

- Reference to Matter Specification:

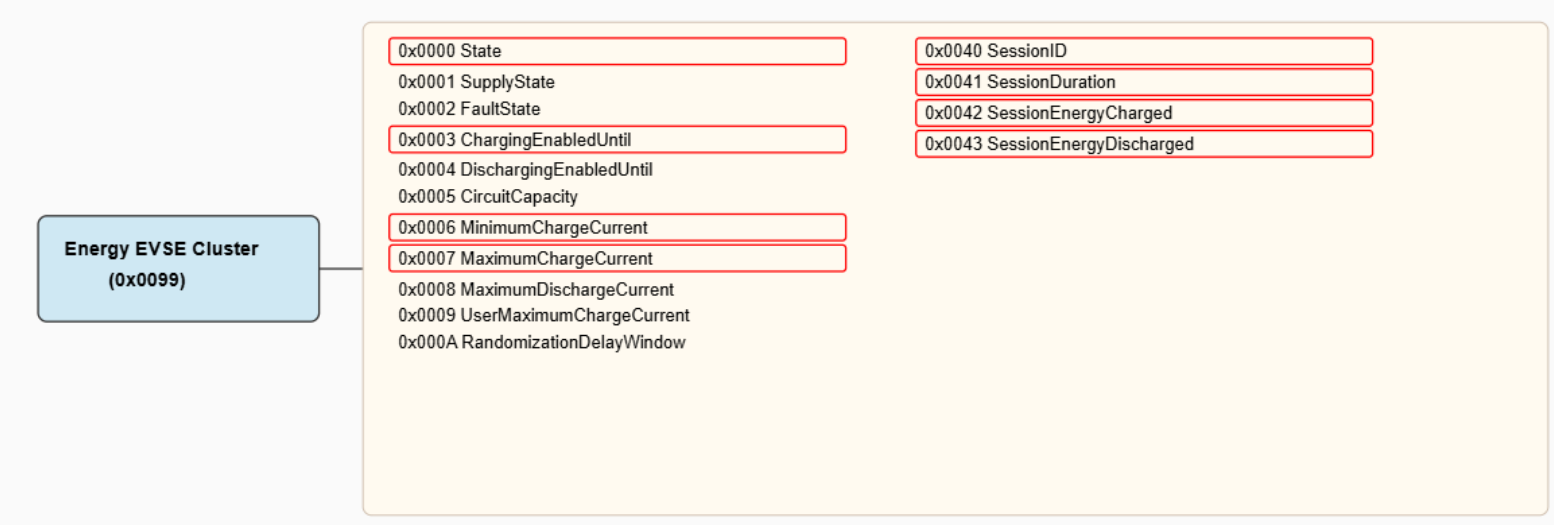
https://handbook.buildwithmatter.com/specification/



NXP Public

Matter Energy EVSE cluster: attribute

- RW612 Matter EVSE device PoC enables the below attributes in the Energy EVSE cluster
  - **State:** Current operational state of the EVSE
  - **ChargingEnabledUntil:** Time until charging is allowed
  - **MinimumChargeCurrent:** Minimum safe charging current
  - **MaximumChargeCurrent:** Maximum allowed charging current. This limit is either set for grid protection or build protection
  - **SessionID:** Unique charging session identifier
  - **SessionDuration:** Charging session duration
  - **SessionEnergyCharged:** Energy delivered to EV



- Reference to Matter Specification:

https://handbook.buildwithmatter.com/specification/

NXP Public

Matter Energy EVSE cluster: state attribute mapping

Below table reflects the EV states as perJ1772 and corresponding Matter attribute mapping implemented in the RW612 EVSE PoC

J1772 CP States	Meaning	Matter Attributes
State A	EV not connected to EVSE	NotPluggedIn
State B	EV connected, not requesting current	PluggedInNoDemand
State C1/D1	EV requests charging, but EVSE not allowing current	PluggedInDemand
State C2/D2	Charging in progress, current flowing	PluggedInCharging
State F/E	Fault condition exists	Fault

NXP Public

Matter Energy EVSE cluster: Matter control commands

- Below are the Matter commands that are enabled in the RW612 EVSE PoC

Matter Commands	Meaning	Command
Disable	Stop the charging to the EV	$ chip-tool energyevse disable 1 1 --timedInteractionTimeoutMs 5000
EnableCharging	Start the charging to the EV	$ chip-tool energyevse enable-charging 10000 6 10 1 1 --timedInteractionTimeoutMs 5000

- Reference to Matter Specification:

https://handbook.buildwithmatter.com/specification/

NXP Public

10

Matter EVSE workflow

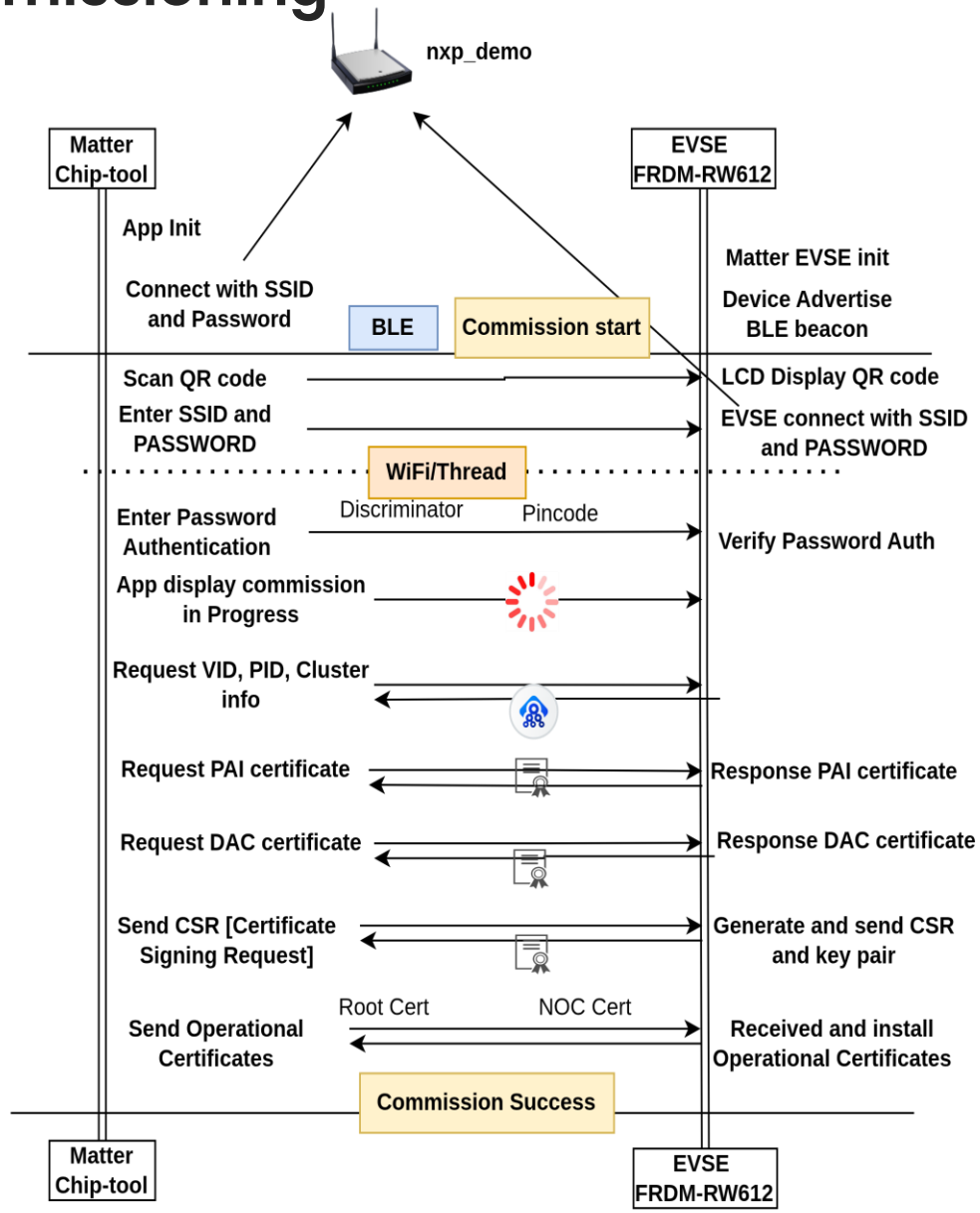


Matter EVSE system workflow: Matter commissioning

- RW612 EVSE PoC enables
  - Matter paring information over QR code:
  - Matter commissioning over BLE
  - Matter control path
    - Matter over Wi-Fi
    - Matter over Thread
  - Matter certificates are enabled via EL2Go



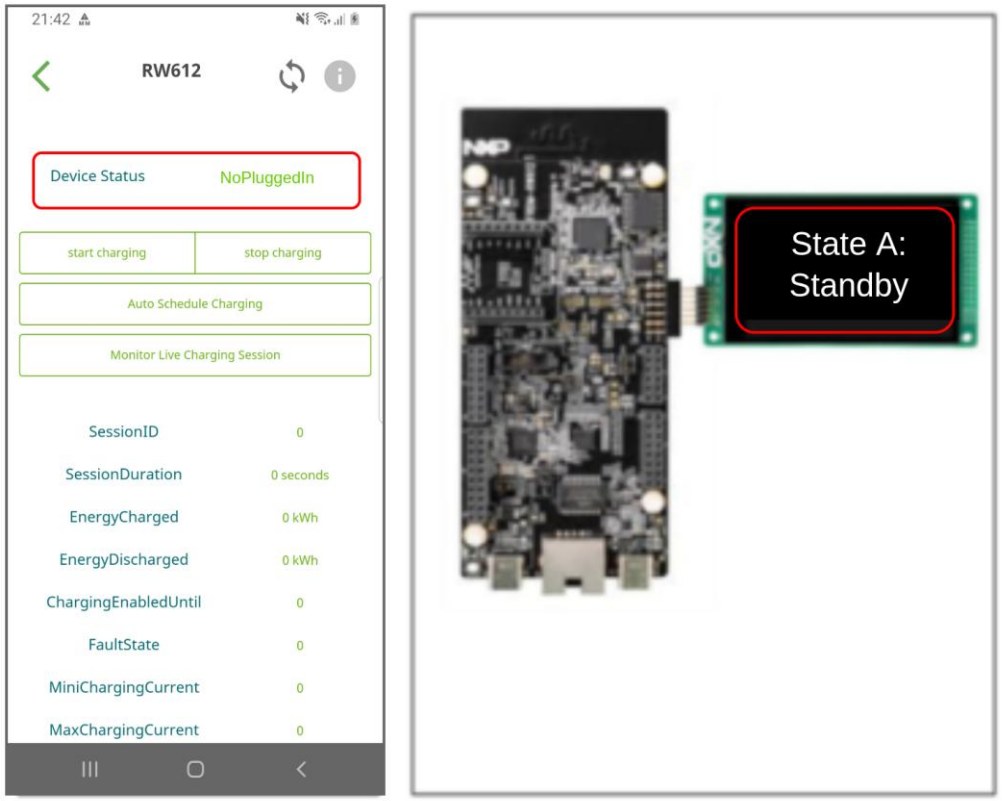


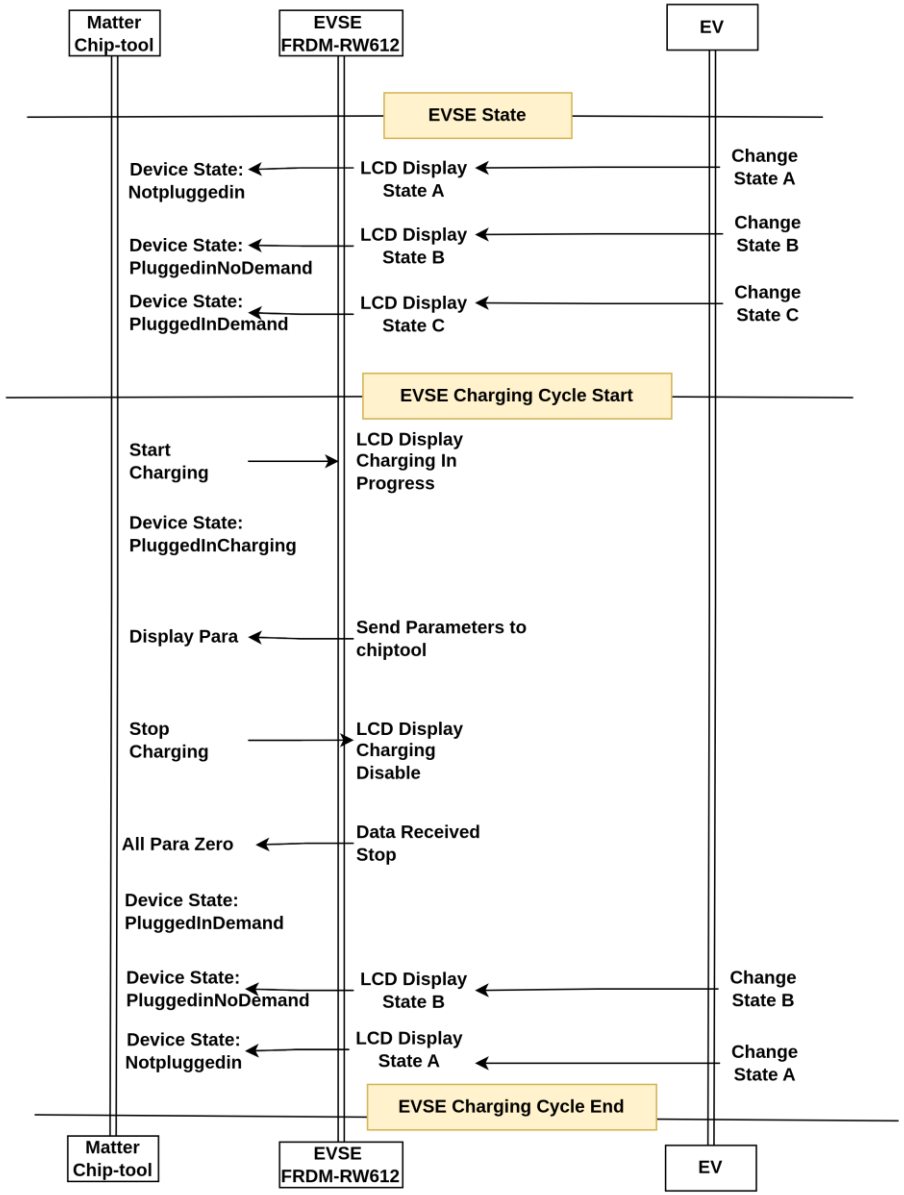


NXP Public

Matter EVSE system workflow: EVSE States

- When EV is not connected to EVSE, EVSE will be in J1772 state A
- State A corresponds to **NoPluggedIn** Matter state

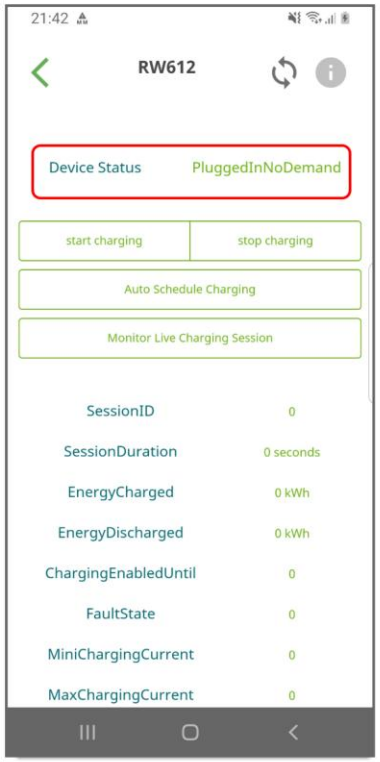


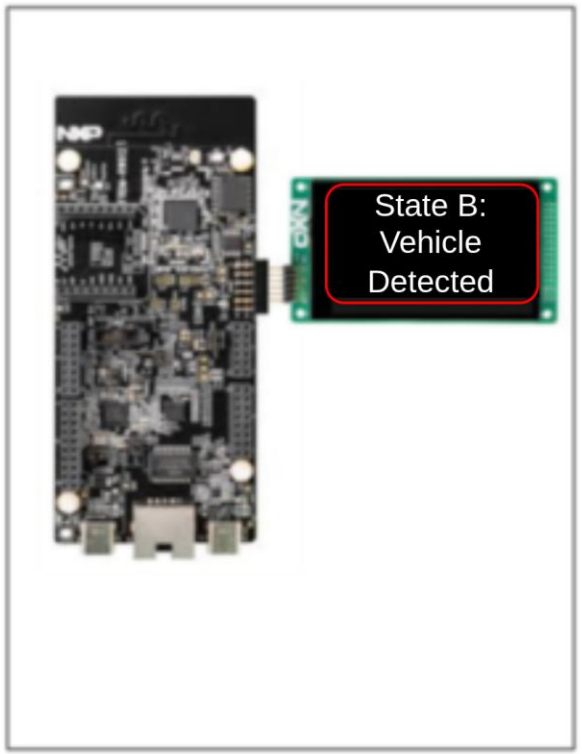


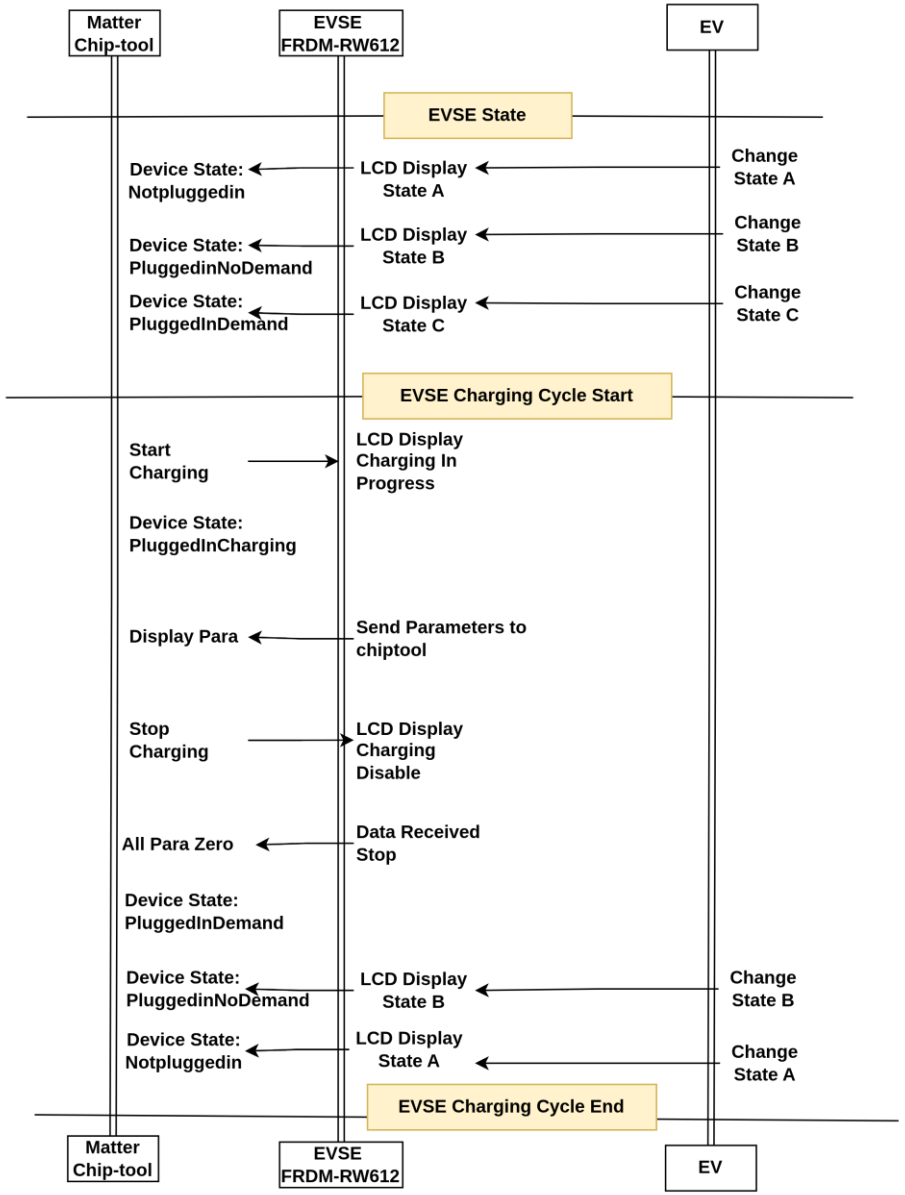
NXP Public

Matter EVSE system workflow: EVSE States

- When EV is connected but not requesting current, EVSE will be in J1772 state B
- State B corresponds to **PluggedInNoDemand** Matter state



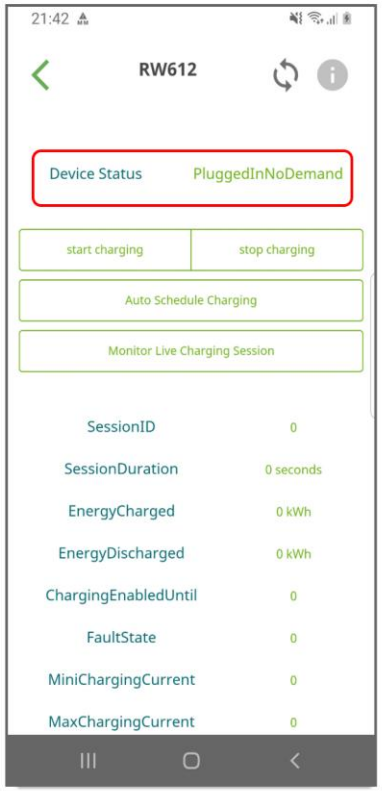


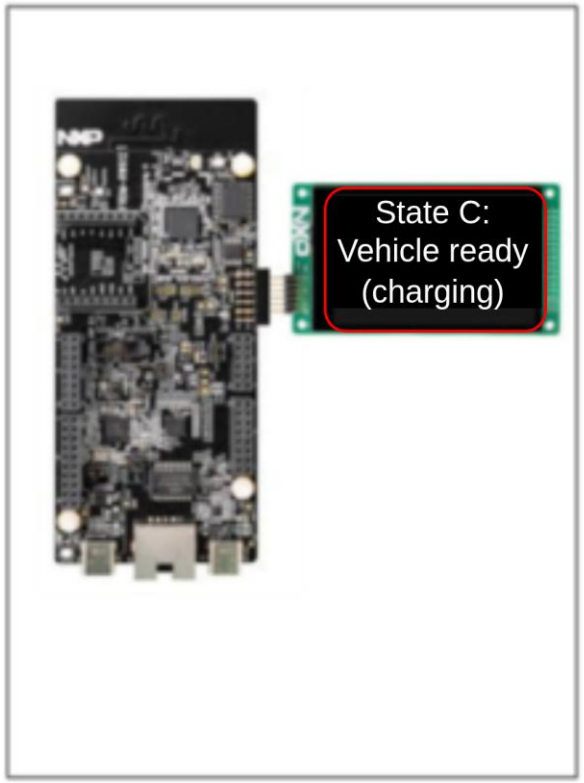


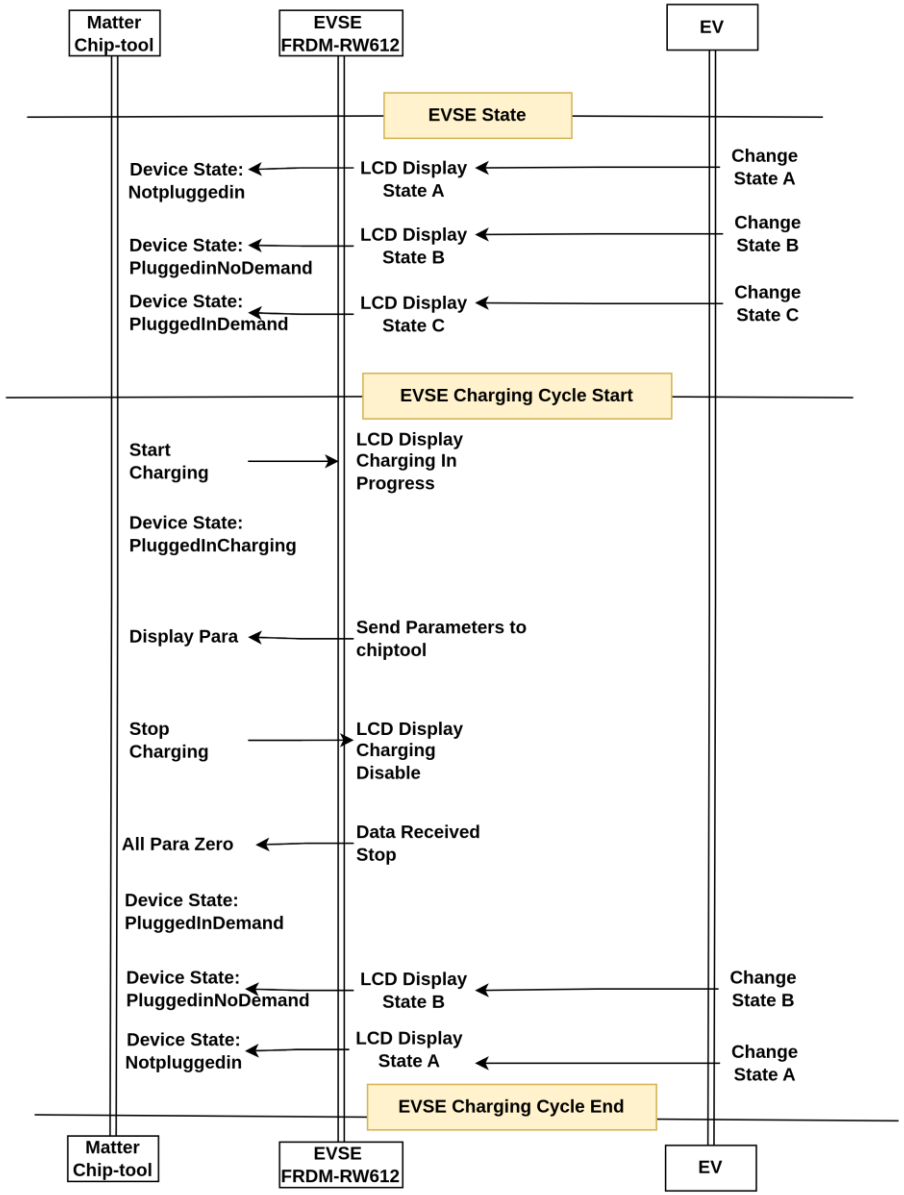
NXP Public

Matter EVSE system workflow: EVSE States

- EV requests charging, but EVSE not allowing current, EVSE will be in J1772 state C1
- State C1 corresponds to **PluggedInDemand** Matter state



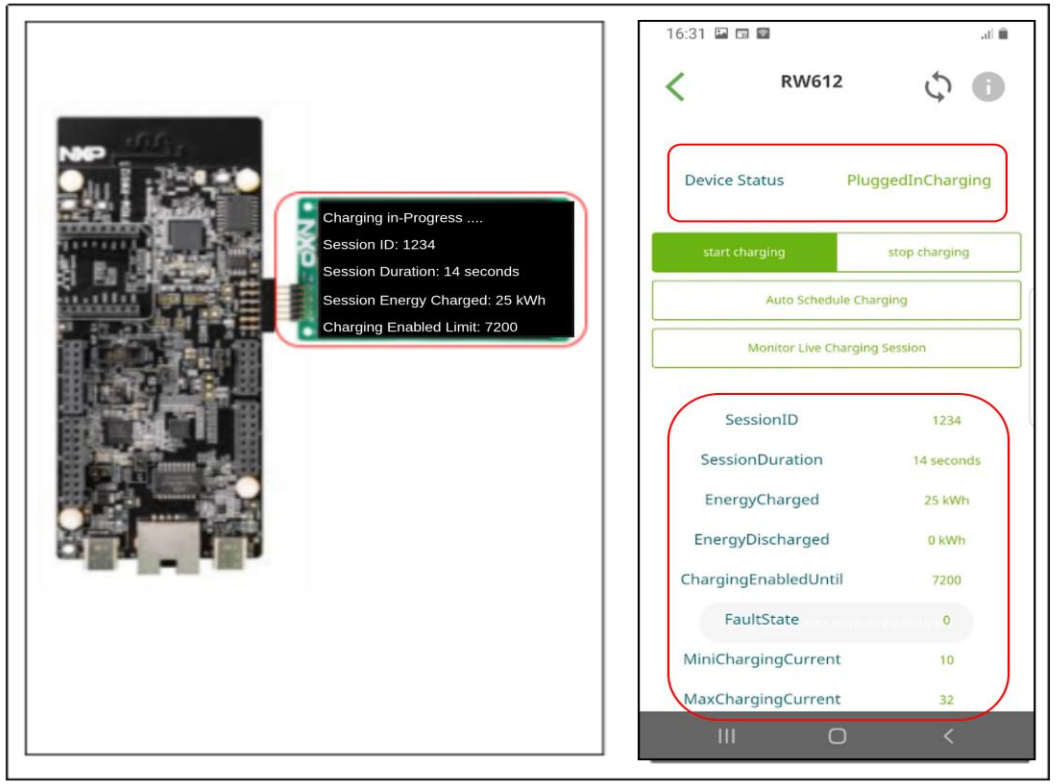


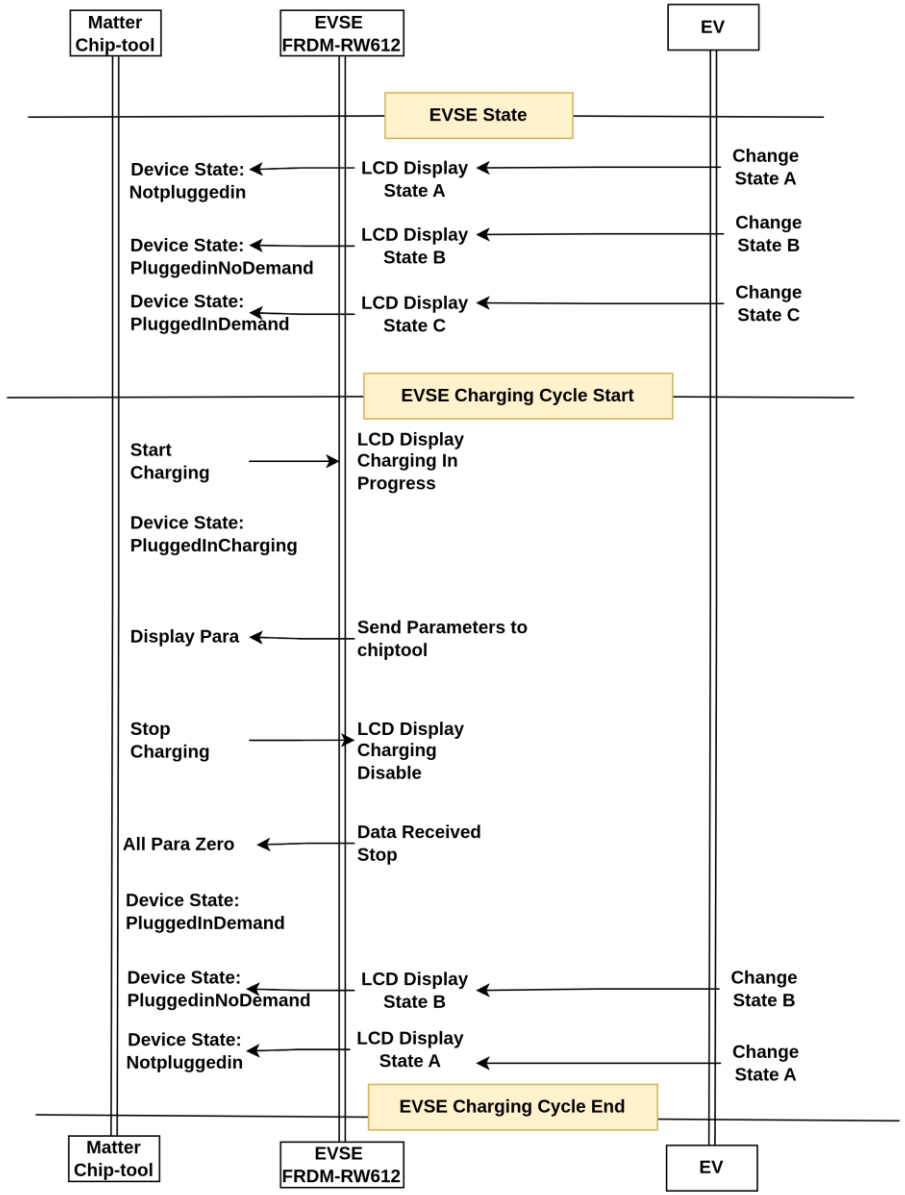


NXP Public

Matter EVSE system workflow: charging cycle

- Charging in progress, current flowing EVSE will be in J1772 state C2
- State C2 corresponds to **PluggedInCharging** Matter state

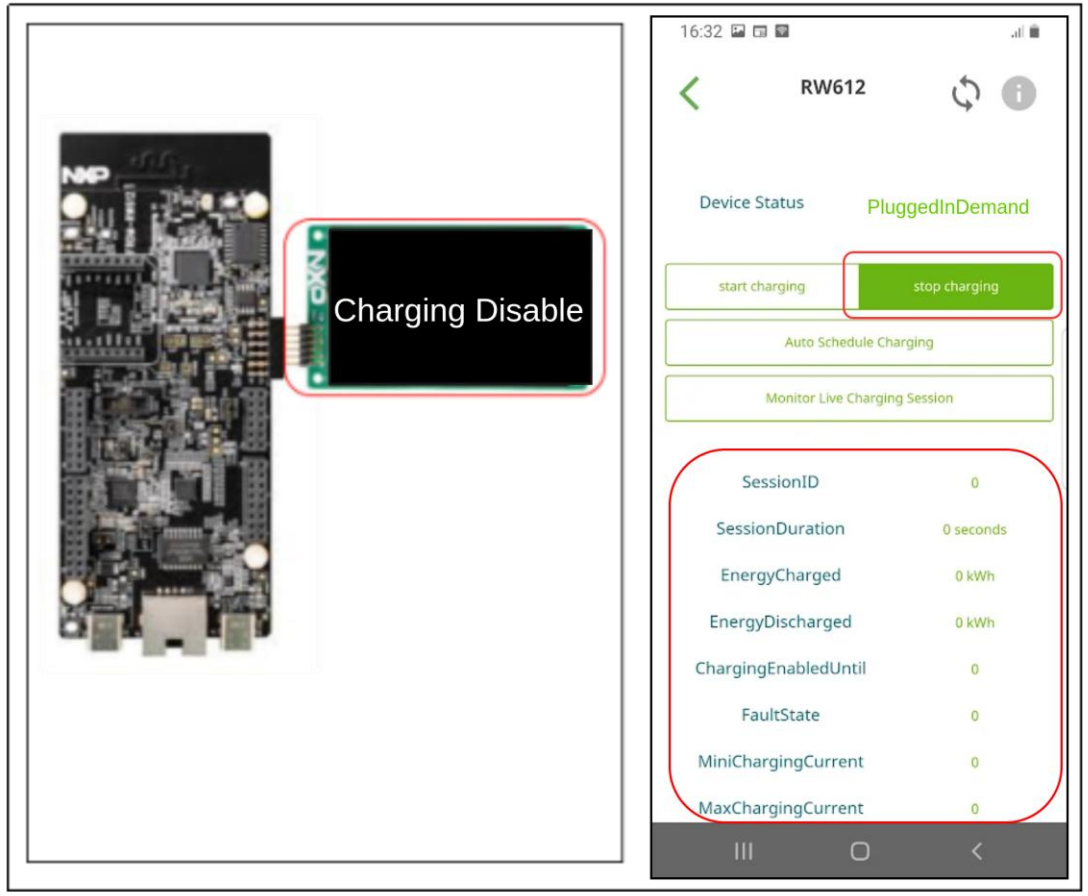


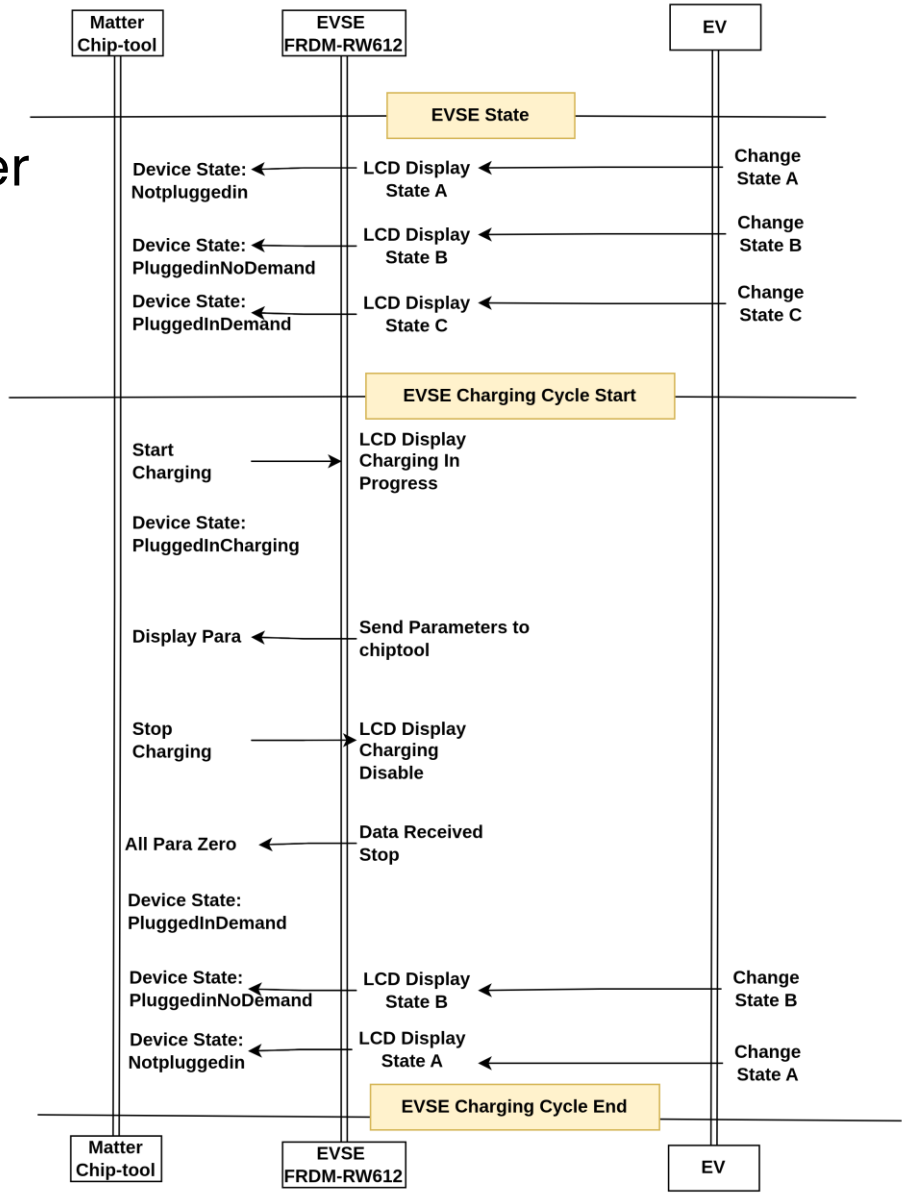


NXP Public

Matter EVSE system workflow: charging cycle

- Charging disable, stop current flowing EVSE will be in J1772 state C1
- State C1 corresponds to **PluggedInDemand** Matter state





NXP Public

Matter Commands

- Below table represents the chip-tool Matter commands defined by Matter specification
- NXP Matter Android app is using the same commands for each of the corresponding actions

CHIPtool command	Meaning
$ chip-tool pairing ble-wifi 1 NXP_Demo 123456789 20200221 3226	Pairs the Matter EVSE device with the controller over BLE and Wi-Fi. • 1 → Node ID assigned to the device • NXP_Demo → Wi-Fi SSID • 123456789 → Wi-Fi password • 20200221 → Setup PIN code for commissioning • 3226 → Discriminator used to identify the device during commissioning
$ chip-tool pairing ble-thread 1 hex: 0e080000000000100004a03000014350600040 01fffe00708 fd5e8c006b5c0cbf0c0402a0f7f8000300001401021234020 8 beef00beef081172030d4d61747465722d4e58502d39330510 00112233445566778899aabbcc7211080410e20d87bb61176d 894ddd9002123e983c 20202021 3226	Pairs the Matter EVSE device with the controller over BLE and Thread. • 1 → Node ID assigned to the device • 20200221 → Setup PIN code for commissioning • 3226 → Discriminator used to identify the device during commissioning • The long hex string is the **Thread operational dataset**.

NXP Public

Matter Commands

- Below table represents the chip-tool Matter commands defined by Matter specification
- NXP Matter Android app is using the same commands for each of the corresponding actions

CHIPtool command	Correct explanation
$ chip-tool energyevse enable-charging 10000 6 10 1 1 --timedInteractionTimeoutMs 5000	Starts the EVSE charging process when the EV is in State C (ready for charging). • 10000 → Charging enabled until (epoch duration / ms) • 6 → MinimumChargeCurrent (mA) • 10 → MaximumChargeCurrent (mA) • 1 1 → Endpoint and cluster IDs • --timedInteractionTimeoutMs 5000 → Timeout for the interaction (5 seconds)
$ chip-tool energyreves read state 1 1	Reads the current EVSE energy state attribute from the commissioned device. • energyreves → Cluster name (Energy Reverse/EVSE cluster) • read state → Command to fetch the current state • 1 1 → Refers to endpoint and cluster IDs (device-specific)
$ chip-tool energyevse read session-id 1 1	Reads the Session ID of the current charging session. Useful for tracking or logging charging events.
$ chip-tool energyevse read session-duration 1 1	Reads the duration of the ongoing charging session.
$ chip-tool energyevse read session-energy-charged 1 1	Reads the total energy charged during the session (typically in Wh or kWh).
$ chip-tool energyevse disable 1 1 --timedInteractionTimeoutMs 5000	Stops the charging process after completion. • 1 1 → Endpoint and cluster IDs • --timedInteractionTimeoutMs 5000 → Timeout for the stop command (5 seconds)

NXP Public

11

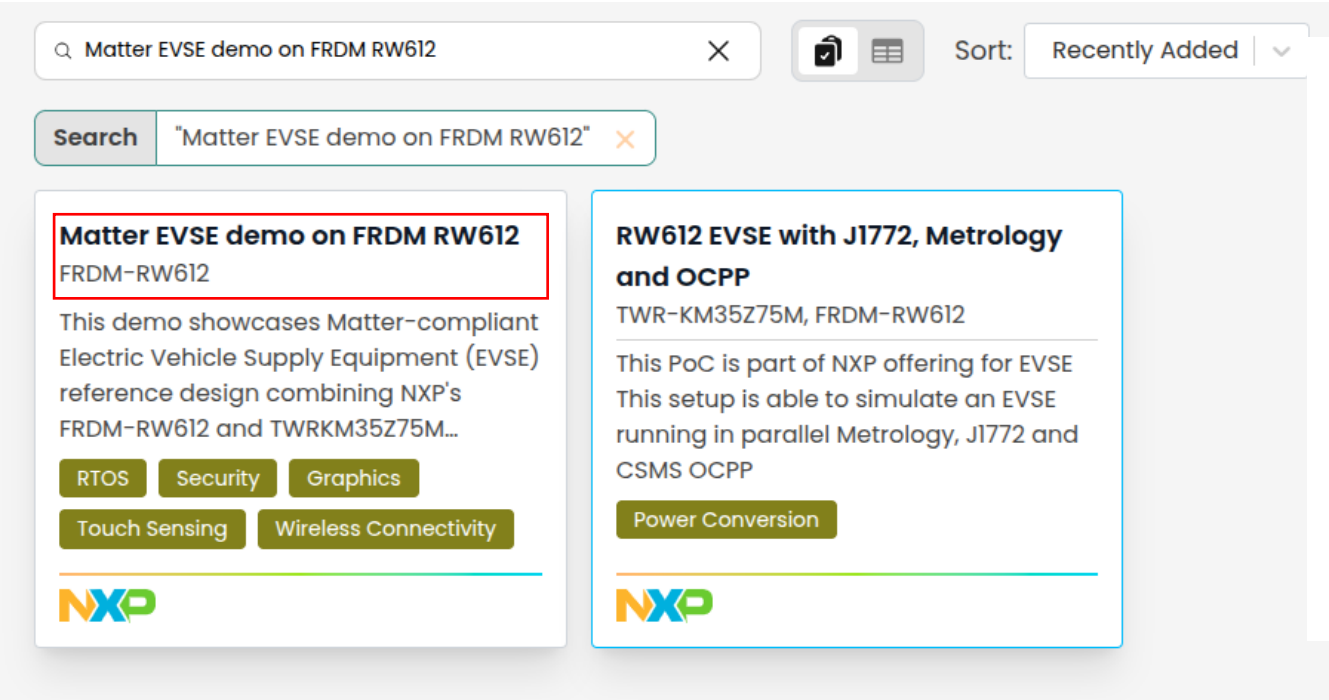
Software details

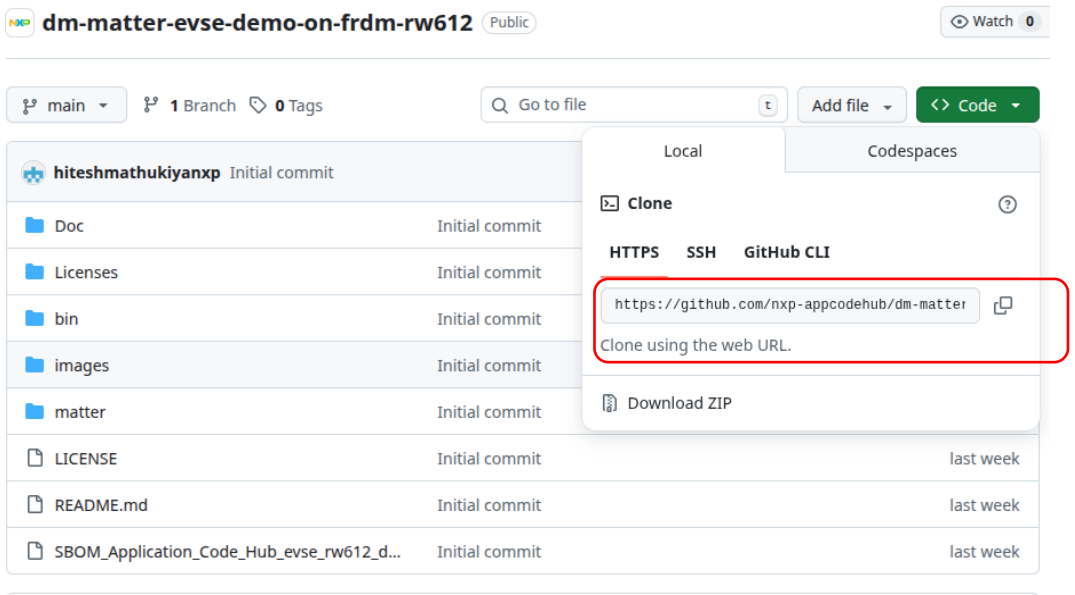


Application Code HUB: EVSE RW612 PoC

- Use below link to get Matter EVSE source code and README.MD

  https://mcuxpresso.nxp.com/appcodehub?search=Matter+EVSE+demo+on+FRDM+RW612

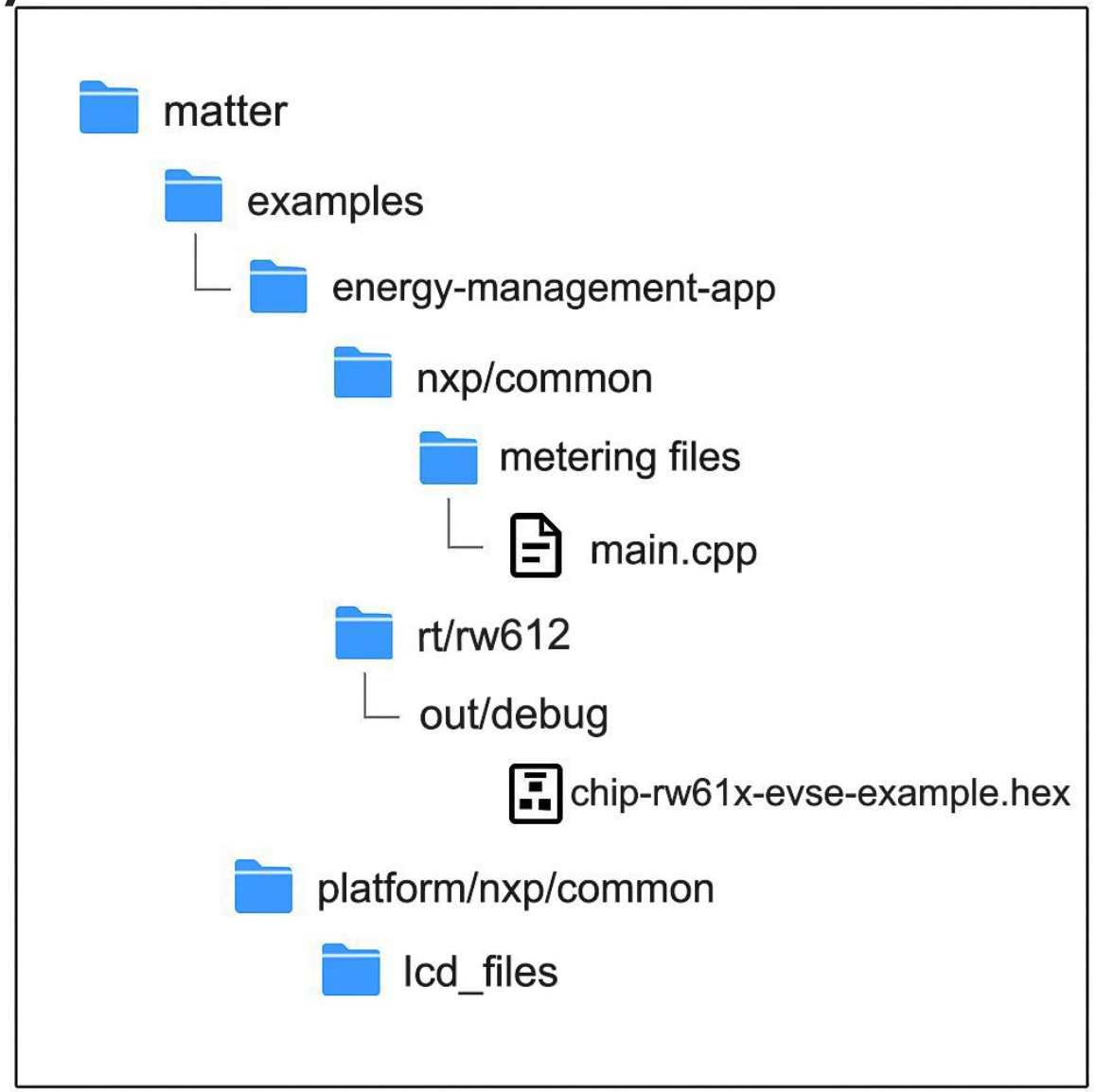




NXP Public

Matter RW612 EVSE Application: Repository structure

- " matter" folder:
  - Contains the energy-management-app example with source code and project build files.
- "engery-management-app" folder:
  - source and include files
  - contains the necessary project build
- "lcd_files" folder:
  - Contains the necessary source and include files for the LVGL display functionality



NXP Public

Software build set-up [build commands]

- Make sure that below prerequisites are correctly installed

```
$ sudo apt-get install git gcc g++ pkg-config libssl-dev libdbus-1-dev \
libglib2.0-dev libavahi-client-dev ninja-build python3-venv python3-dev \
python3-pip unzip libgirepository1.0-dev libcairo2-dev libreadline-dev
```

- Step 1: Clone Matter repo

```
$ git clone https://github.com/NXP/matter.git
$ cd matter
$ git fetch --all --tags
$ git checkout release/v1.4.2
$ scripts/checkout_submodules.py --shallow --platform nxp --recursive
```

- Step 2: Clone EVSE repo

```
$  git clone https://github.com/nxp-appcodehub/dm-matter-evse-demo-on-frdm-
rw612.git
```

- Step 3: Overwrite the files

```
- Folders list and path that needs to place,

1. energy-management-app  -->  matter/examples/
2. lcd_files --> matter/examples/platform/nxp/common/
3. nxp_matter_support --> matter/third_party/nxp/nxp_matter_support
4. energy-evse-server --> matter/src/app/clusters/

- Files list and path that needs to be place,

1. CommissioningWindowManager.cpp --> matter/src/app/server
2. Server.cpp  --> matter/src/app/server
3. CommonDeviceCallbacks.cpp -->
matter/examples/platform/nxp/common/device_callbacks/source/
4. AppCLIBase.cpp -->
matter/examples/platform/nxp/common/matter_cli/source/AppCLIBase.cpp
```

- Step 4: Activate local environment

```
$ cd matter/
$ source scripts/bootstrap.sh
$ source scripts/activate.sh
```

- Step 5: Download the SDK 25.03

```
$ third_party/nxp/nxp_matter_support/scripts/update_nxp_sdk.py --platform
common --force

- Step 5.1:

After Download SDK, add below macro at line 225 in "rw61x.gni" file at path
"matter/third_party/nxp/nxp_matter_support/gn_build/rt_sdk/rw61x/rw61x.gni"

defines += ["CONFIG_PRIVATE_ECC_KEY_BLOB_SIZE=48"]
```

- Step 6: Go to example and build Matter+WiFi

```
$ cd examples/energy-management-app/nxp/rt/rw61x/
$ gn gen --args="chip_enable_wifi=true board_version=\"frdm\"
nxp_enable_matter_cli=true" out/debug
$ ninja -C out/debug
```

- Step 7: Rename .hex to .srec and Flash on FRDM-RW612

```
$ cd out/debug
$ cp chip-rw61x-evse-example.hex chip-rw61x-evse-example.srec
```

- Step 8: Now, flash “chip-rw61x-evse-example.srec” using MCUXpresso on FRDM-RW612 at address 0x8000400

NXP Public

Matter RW612 EVSE Application: Build instructions

Description	Commands / Actions
Prerequisites – Install required packages	$ sudo apt-get install git gcc g++ pkg-config libssl-dev libdbus-1-dev libglib2.0-dev libavahi-client-dev ninja-build python3-venv python3-dev python3-pip unzip libgirepository1.0-dev libcairo2-dev libreadline-dev
Step 1: Clone Matter repo	$ git clone https://github.com/NXP/matter.git cd matter git fetch --all --tags git checkout release/V1.4.2 scripts/checkout_submodules.py --shallow --platform nxp --recursive
Step 2: Clone EVSE repo	$ git clone https://github.com/nxp-appcodehub/dm-matter-evse-demo-on-frdm-rw612.git
Step 3: Overwrite the files/folders	Place folders: 1. energy-management-app → matter/examples/ 2. lcd-files → matter/examples/platform/nxp/common/ 3. nxp_matter_support → matter/third_party/nxp/nxp_matter_support 4. energy-evse-server → matter/src/app/clusters/ Place files: 1. CommissioningWindowManager.cpp → matter/src/app/server 2. Server.cpp → matter/src/app/server 3. CommonDeviceCallbacks.cpp → matter/examples/platform/nxp/common/device_callbacks/source/ 4. AppCLIBase.cpp → matter/examples/platform/nxp/common/cli/source/

NXP Public

Matter RW612 EVSE Application: Build instructions

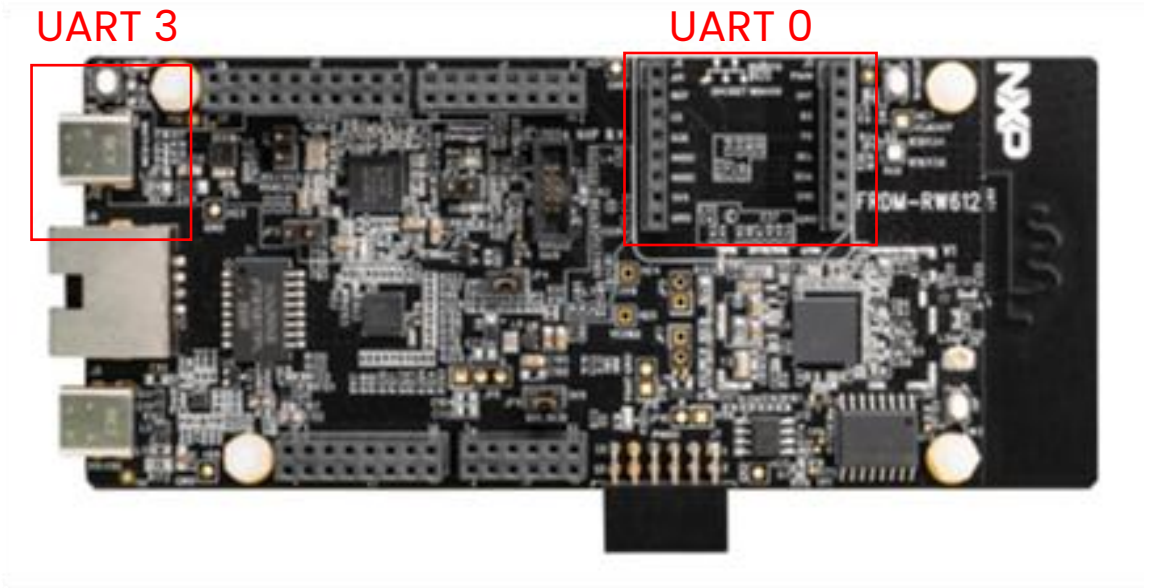
Description	Commands/actions
Step 4: Activate local environment	$ cd matter/ source scripts/bootstrap.sh source scripts/activate.sh
Step 5: Download SDK	$ third_party/nxp/nxp_matter_support/scripts/update_nxp_sdk.py --platform common --force
Step 6 A: Build Matter-WiFi	$ cd examples/energy-management-app/nxp/rt/rw61x/ gn gen --args="chip_enable_wifi=true board_version=\"frdm\" nxp_enable_matter_cli=true" out/debug ninja -C out/debug
Step 6 B: Build Matter-Thread	$ gn gen --args="board_version=\"frdm\" nxp_enable_matter_cli=true chip_enable_openthread=truechip_inet_config_enable_ipv4=false chip_config_network_layer_ble=true" out/debug
Step 7: Rename .hex to .srec [optional]	$ cd out/debug cp chip-rw61x-evse-example.hex chip-rw61x-evse-example.srec
Step 8: Flash on FRDM-RW612	Flash chip-rw61x-evse-example.srec using MCUXpresso

Note: Customer can either choose step 6A or 6B based on Matter over Wi-Fi or Open Thread

NXP Public

Matter EVSE RW61x: Logging/command console

- Current PoC enables both logging and Command console on UART3
- As per customer needs, this can be use same/different UART ports
  - File: third_party/nxp/nxp_matter_support/examples/platform/rt/nxp_sdk_reconfig.cmake
  - Same UART port
    - Command and Logs on UART 3: set this flag in the above file "DEBUG_CONSOLE_UART_INDEX = 3“
  - Different UART port for
    - Command on UART 3 and Logs on UART 0: set this flag in the above file "DEBUG_CONSOLE_UART_INDEX = 0"




```
exit            Exit the shell application
help            List out all top level commands
version         Output the software version
ble             Bluetooth LE commands
wifi            Wi-Fi commands
config          Manage device configuration. Usage to dump value: config [param_name] and to set some values (discriminator): config [param_name] [param_value].
device          Device management commands
onboardingcodes Dump device onboarding codes. Usage: onboardingcodes none|softap|ble|onnetwork [qrcode|qrcodeurl|manualpairingcode]
dns             DNS client commands
stat            Statistics commands
echo            Echo back provided inputs
log             Logging utilities
rand            Random number utilities
mattercommissioning Open/close the commissioning window. Usage : mattercommissioning [on|off]
matterfactoryreset Perform a factory reset on the device
matterreset     Reset the device
setenergy       report mocked energy readings: power(mW) voltage(mV) current(mA), example: setenergy 10000 5000 2000
utctime         utctime test, usage: init|sync
```


NXP Public

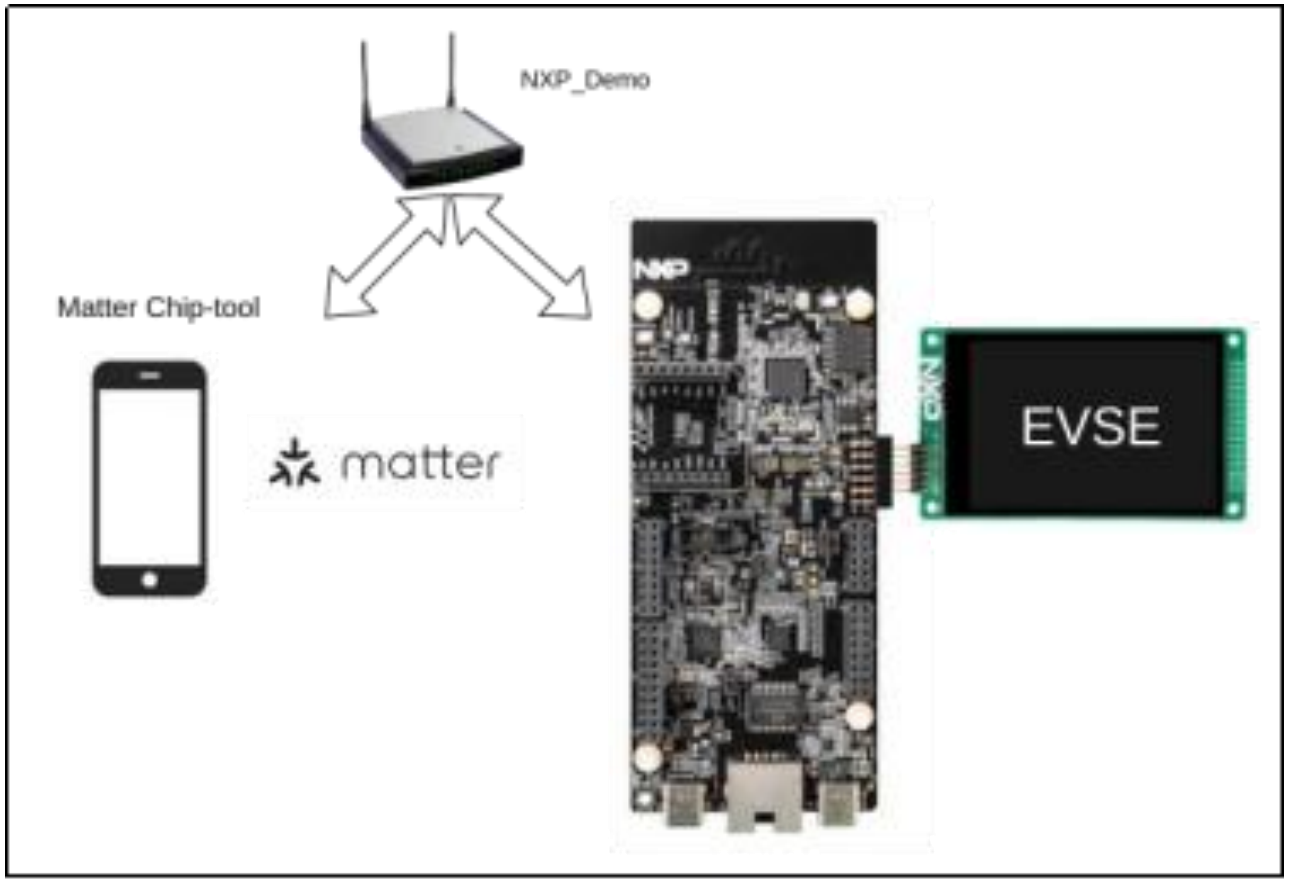
13

Matter EVSE Wi-Fi with Android CHIPtool

Demonstration

Test: commissioning process

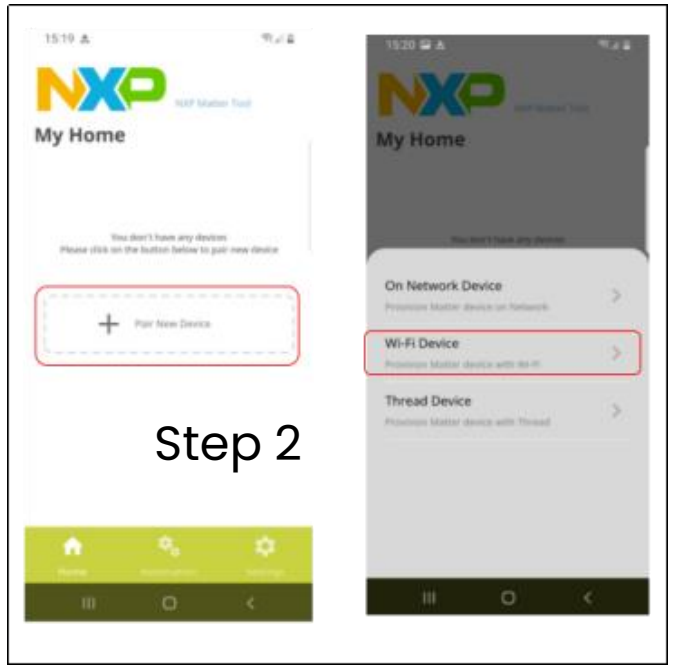
- Matter Wi-Fi Commission Process





Step 1



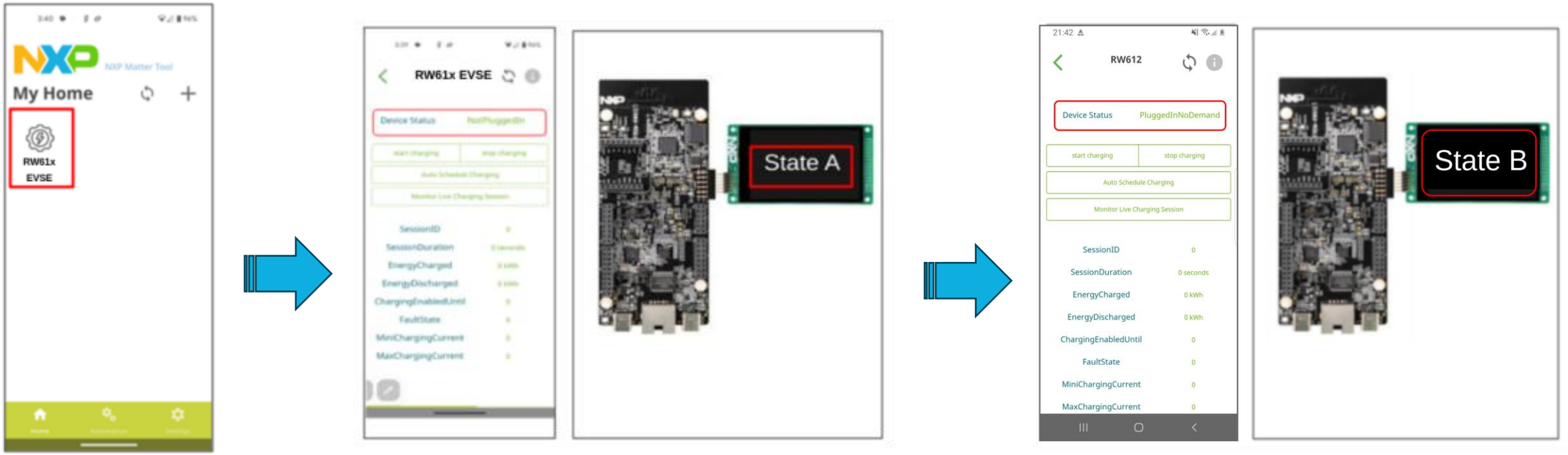


Step 2

NXP Public

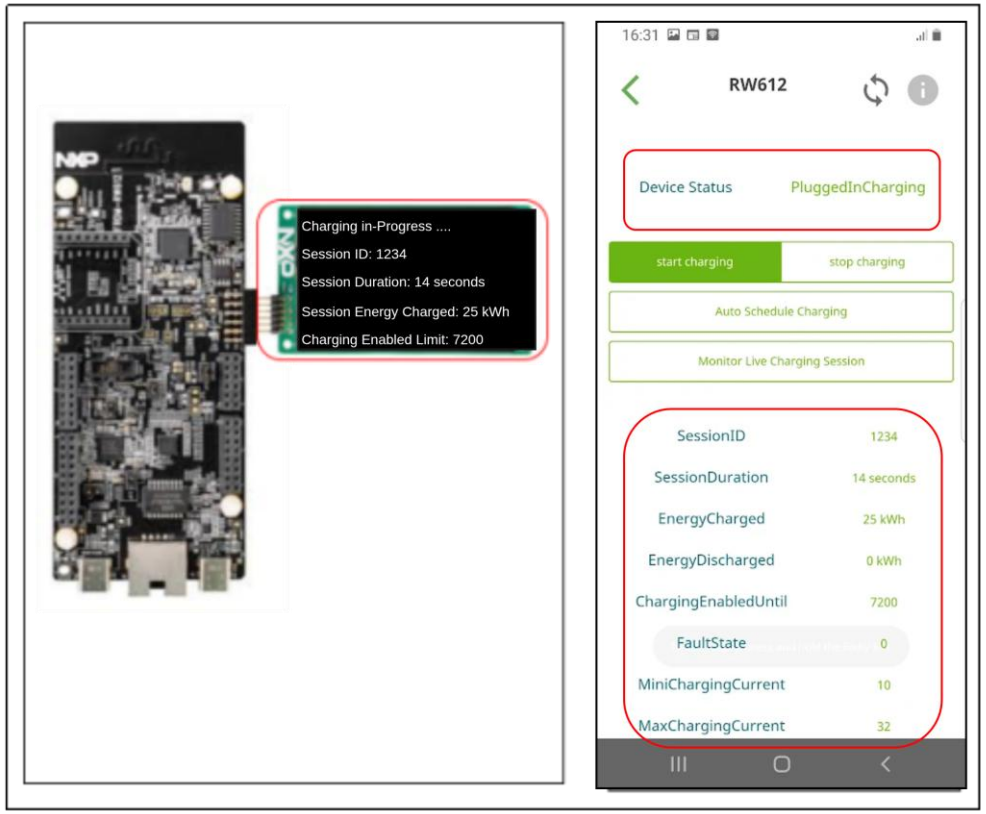
Test: EVSE state

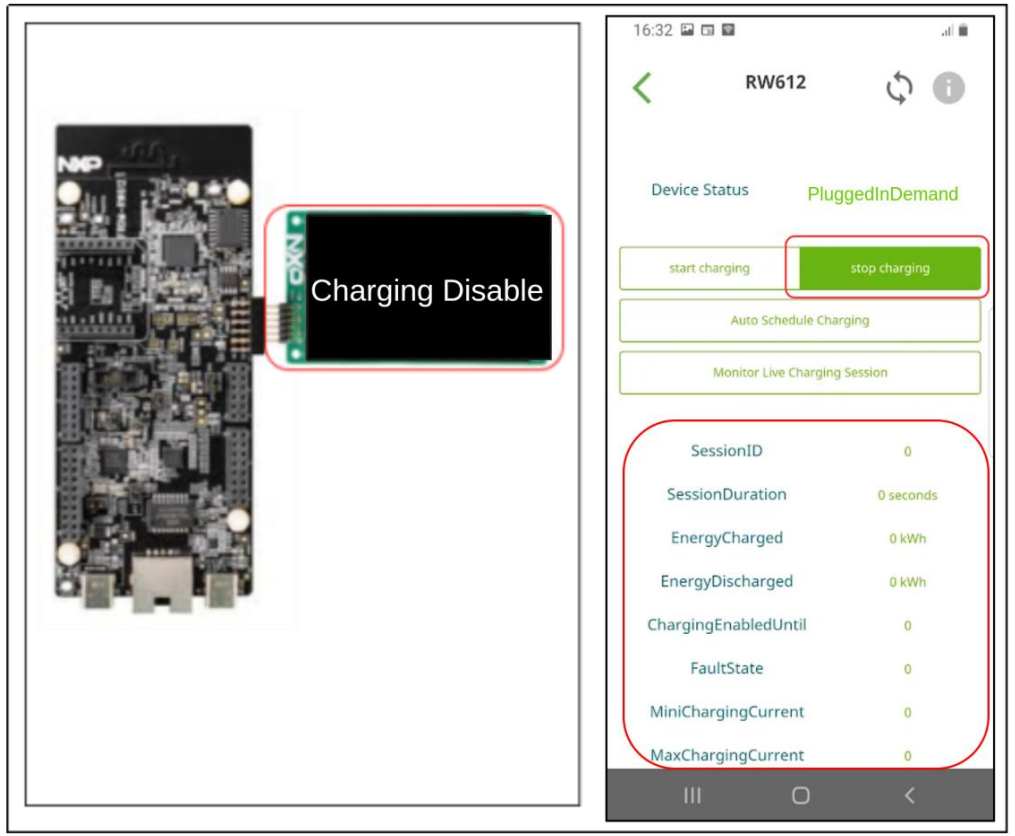
Below image illustrate the EV State transition



NXP Public

Test: charging enable/disable





NXP Public

14

Matter EVSE OpenThread (OT) support



Matter EVSE OpenThread support: build command

- Enter below build command to build EVSE with OT support

```
$ gn gen --args="board_version=\"frdm\" nxp_enable_matter_cli=true chip_enable_openthread=true
chip_inet_config_enable_ipv4=false chip_config_network_layer_ble=true" out/debug
```

- Flash the generated binary onto the FRDM-RW612 board, then open a CLI terminal. Typing help will display the available otcli commands and their usage information.

```
help                Show help
exit                Exit shell
list                List all commands
version             Show version
ble                 BLE commands
wifi                Wi-Fi commands
config              Configuration commands
otbrcmd             OpenThread Border Router command
otcli               Dispatch OpenThread CLI command
echo                Echo input text
log                 Logging commands
matterfactoryreset  Perform Matter factory reset
matterrestart       Restart Matter stack
useramp             User AMP command
utcTimeTest         UTC time test command
```


NXP Public

Matter EVSE OpenThread (OT) with Android CHIPtool



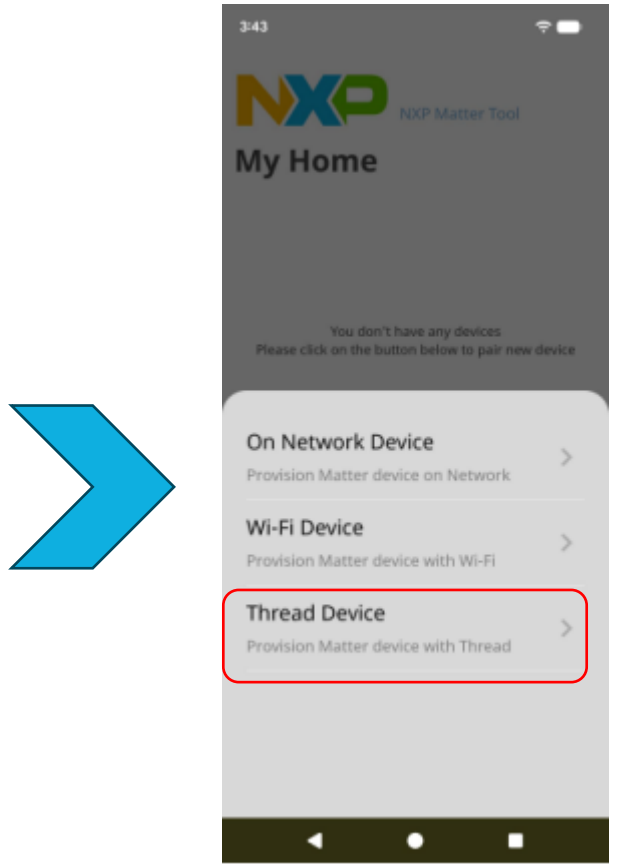
Test 1: Matter EVSE OT commissioning process with Android Matter CHIPtool app

Test1: Matter EVSE OT commissioning Process with Android Matter CHIPtool app

Description	Commands/actions

Step 1: Initialize and activate dataset (OTBR + chip-tool on same Wi-Fi)	$ otcli dataset init new
	$ otcli dataset
	$ otcli dataset commit active
	$ otcli ifconfig up
	$ otcli thread start

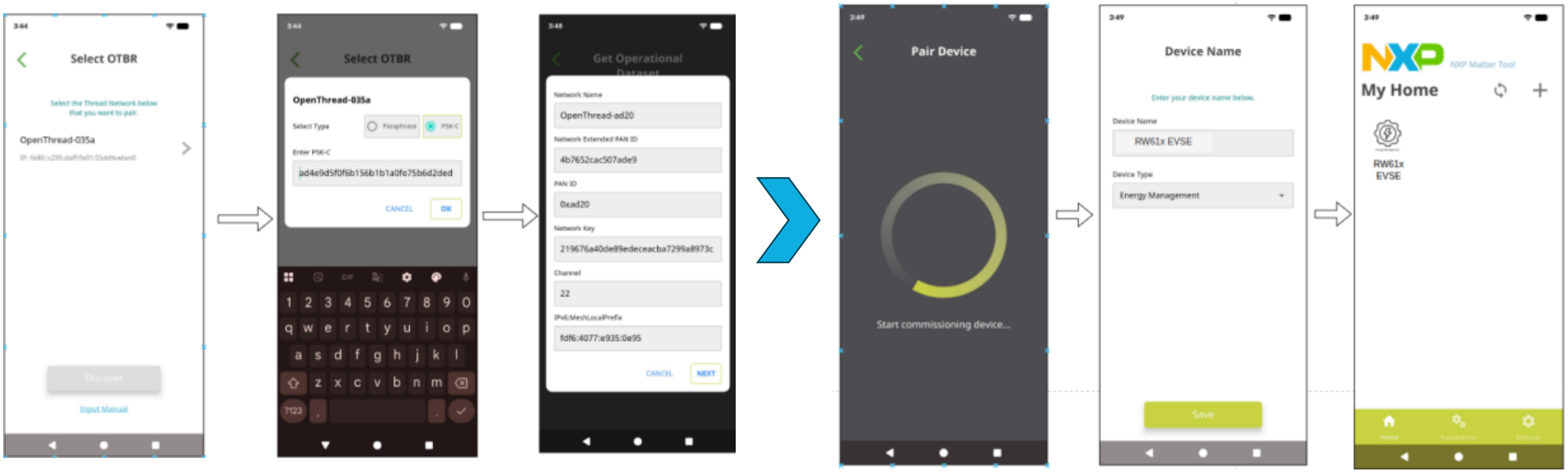
Step 2: Verify leader state (~1 min later)	$ otcli state → should return leader
Step 3: Retrieve PSKc for Android Matter controller	$ otcli pskc → e.g., 198886f519a8fd7c981fee95d72f4ba7
Step 4: Power on EVSE OT Device	Ensure the device advertises for commissioning



NXP Public

Test 1: Matter EVSE OT commissioning process with Android Matter CHIPtool app

After that select OTBR and click on PSK-C and enter 12-digit key and press next.



NXP Public

Matter EVSE OpenThread (OT) with Linux CHIPtool



Test 2: Matter EVSE OT commissioning process with Linux CHIPtool

Description	Commands/actions
Commission device	$ chip-tool pairing ble-thread 301 hex: 0e080000000000100004a03000014350600040 01fffe00708fd 5e8c006b5c0cbf0c0402a0f7f8000300001401021234020 8beef 00beef081172030d4d61747465722d4e58502d393305100011223 3445566778899aabbcc7211080410e20d87bb61176d894ddd90 02123e983c 20202021 3226 - The long hex string is the Thread operational dataset.
Read EV state (initial)	$ chip-tool energyseve read state 301 1
Enable charging (when EV in state C)	$ chip-tool energyseve enable-charging 10000 6 10 301 1 -- timedInteractionTimeoutMs 5000
Read Session ID	$ chip-tool energyseve read session-id 301 1
Read Session Duration	$ chip-tool energyseve read session-duration 301 1
Read Session Energy Charged	$ chip-tool energyseve read session-energy-charged 301 1

NXP Public

16

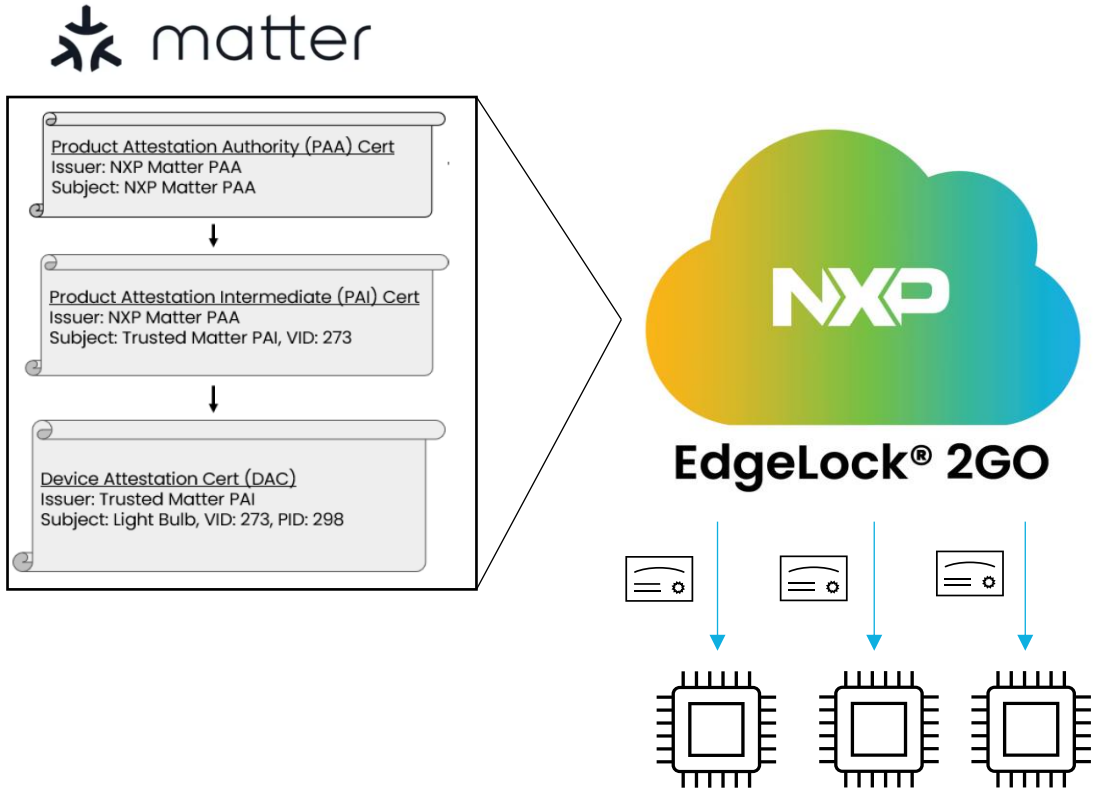
Edgelock2GO

Matter Manufacturing Flow

Matter with EdgeLock2GO

Matter Device Attestation:

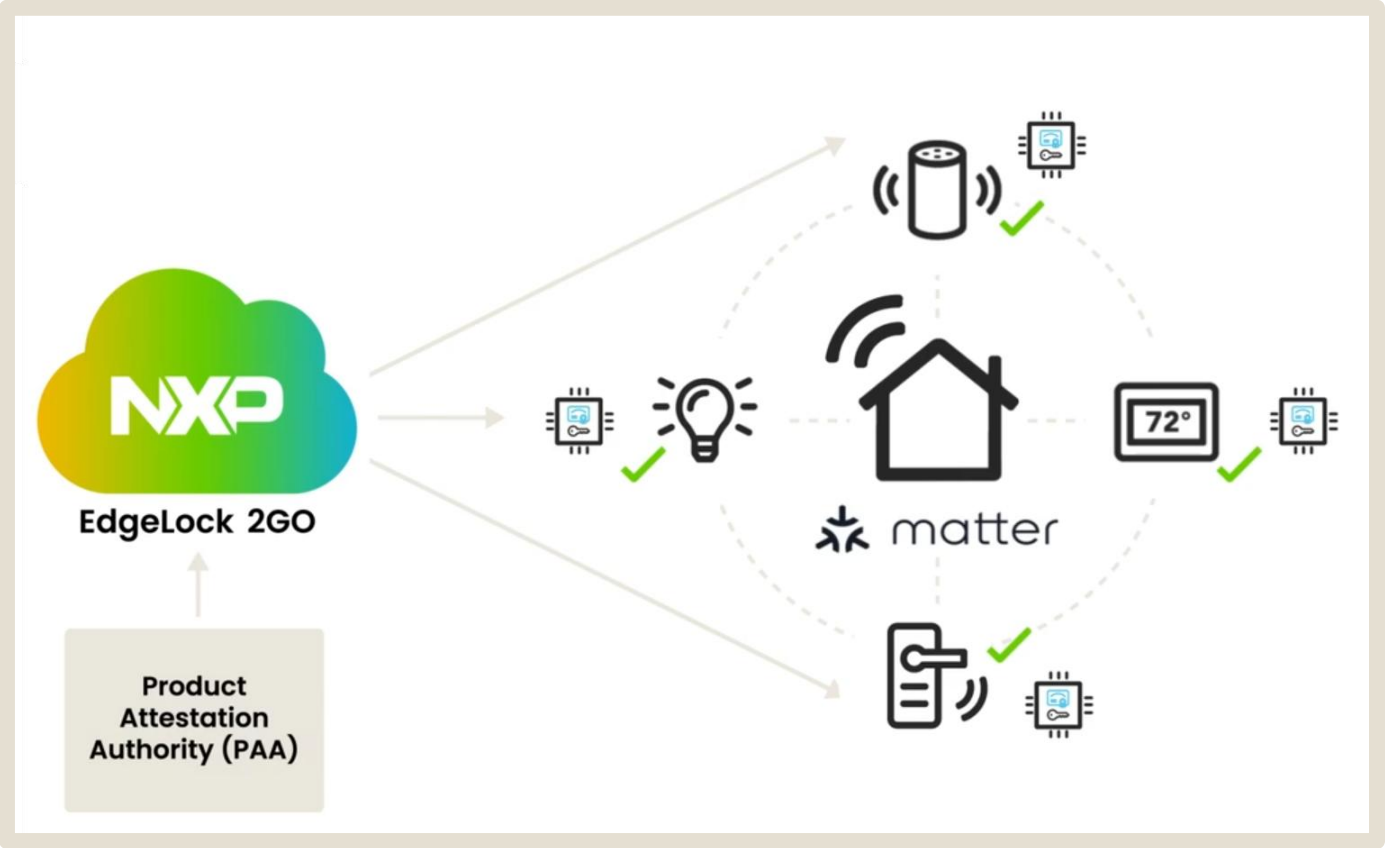
- Each device should have a unique cryptographic identity
- Attestation is based on Public Key infrastructure(PKI) and uses X.509 certificates to authenticate Matter products and device vendors
- All Matter devices are required to use an attestation keypair and an X.509 certificate signed by a Trusted Certificate Authority (CA) - Product Attestation Authority (PAA)



NXP Public

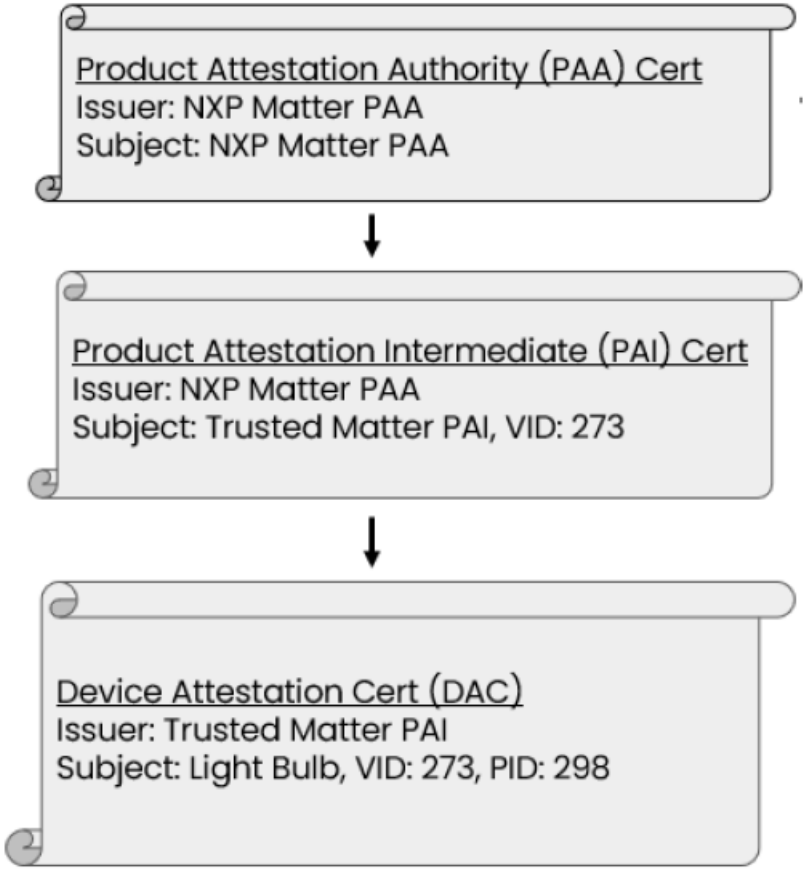
Matter with EdgeLock2GO

- NXP is a Product Attestation Authority approved by the Connectivity Standards Alliance (CSA)
- Leverages NXP security infrastructure
- Leverages the hardware security capabilities of the RW612 device for protecting device keys.
- NXP-generated attestation certificates work with devices that use any MCU or MPU, from NXP or another supplier.
- More information on how to provision RW612 devices with Matter credentials using EdgeLock2GO are available in the Application Note AN14705, on www.nxp.com.



NXP Public

Matter Certificate Chain



PAA (Root)
└── Signs → PAI (Intermediate)
└── Signs → DAC (Device Certificate)

During **Matter commissioning:**

1. EVSE presents its DAC
2. Commissioner verifies DAC signature using PAI public key
3. Commissioner verifies PAI signature using PAA public key
4. Commissioner checks that PAA is in its trusted root store
5. If chain validates, device is authenticated as genuine

NXP Public

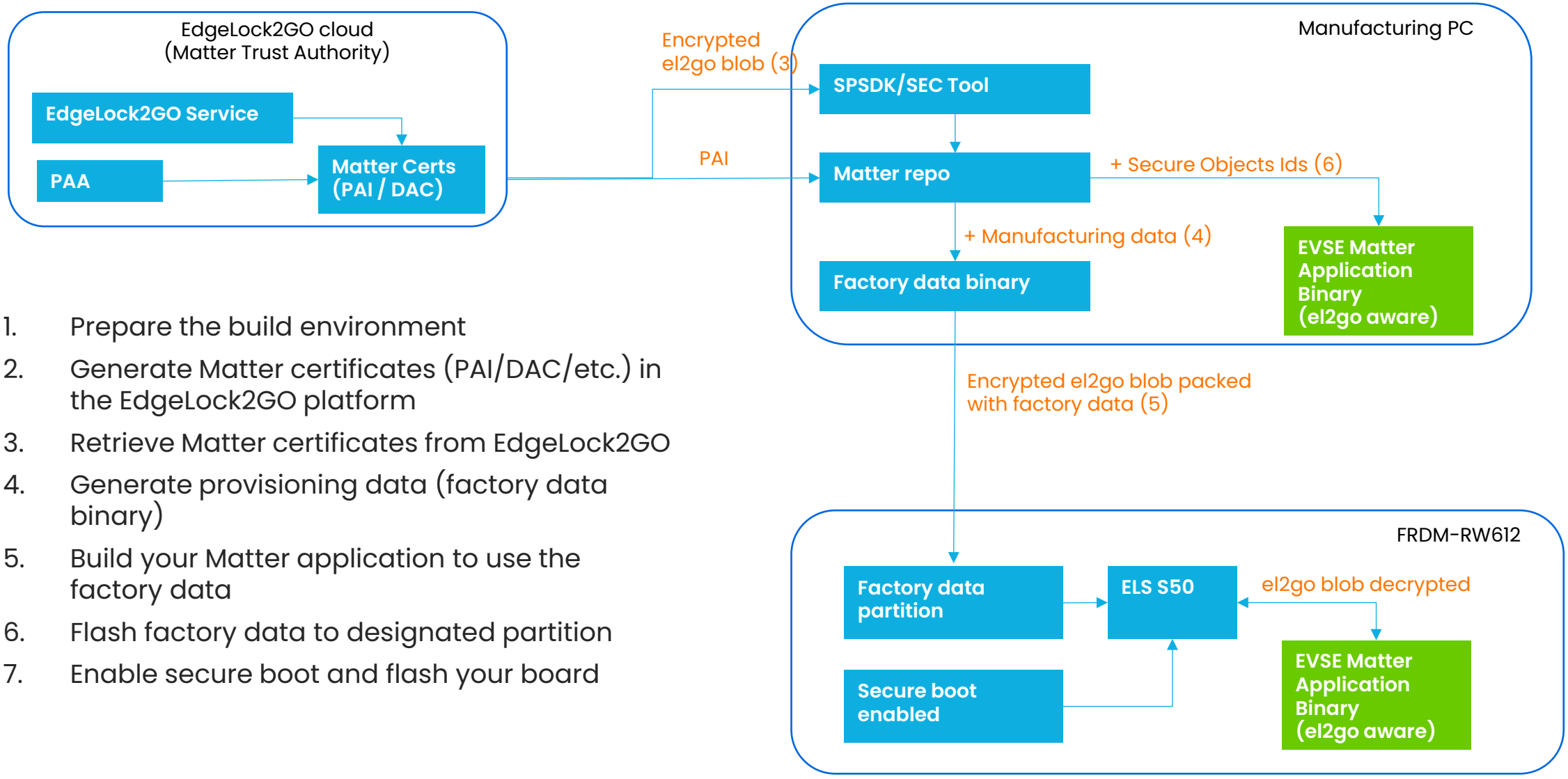
Matter Certificate Chain

Component	Secret?	Integrity protected – how?	Stored securely?
PAA	No	Self-signature + trust anchor	Not mandatory (on commissioner)
PAI	No	PAA signature verification	Not mandatory
DAC certificate	No	PAI signature verification	Not mandatory
DAC private key	Yes	Secure storage + access control	Mandatory

Integrity protection comes from x.509 signature verification. If an attacker modifies the content of the PAI or DAC (Vendor ID, Product ID, Public key, Any extensions), the signature verification will fail.

NXP Public

Matter Manufacturing Flow with EdgeLock2GO



1. Prepare the build environment
2. Generate Matter certificates (PAI/DAC/etc.) in the EdgeLock2GO platform
3. Retrieve Matter certificates from EdgeLock2GO
4. Generate provisioning data (factory data binary)
5. Build your Matter application to use the factory data
6. Flash factory data to designated partition
7. Enable secure boot and flash your board

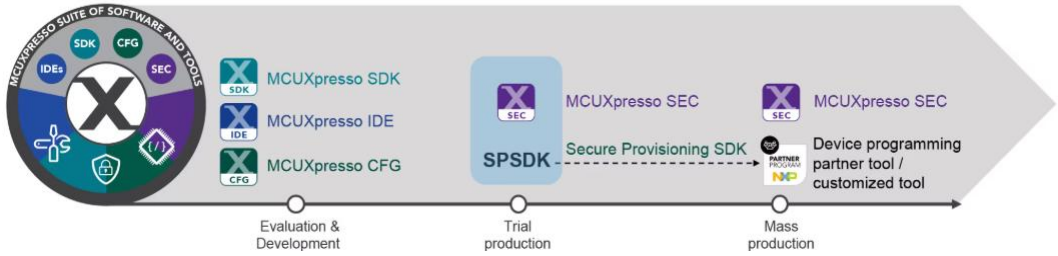
NXP Public

Matter Manufacturing Flow with EdgeLock2GO

Step 1 – Prepare the environment

- Download Matter repository: build the **Matter/CHIP Certificate Tool (chip-cert)** and **SPAKE2P Parameters Tool (spake2p)**
- Install **OpenSSL** or similar cryptographic tool
- Install **NXP Secure Provisioning SDK/MCUXpresso Secure Provisioning Tool** for provisioning

matter



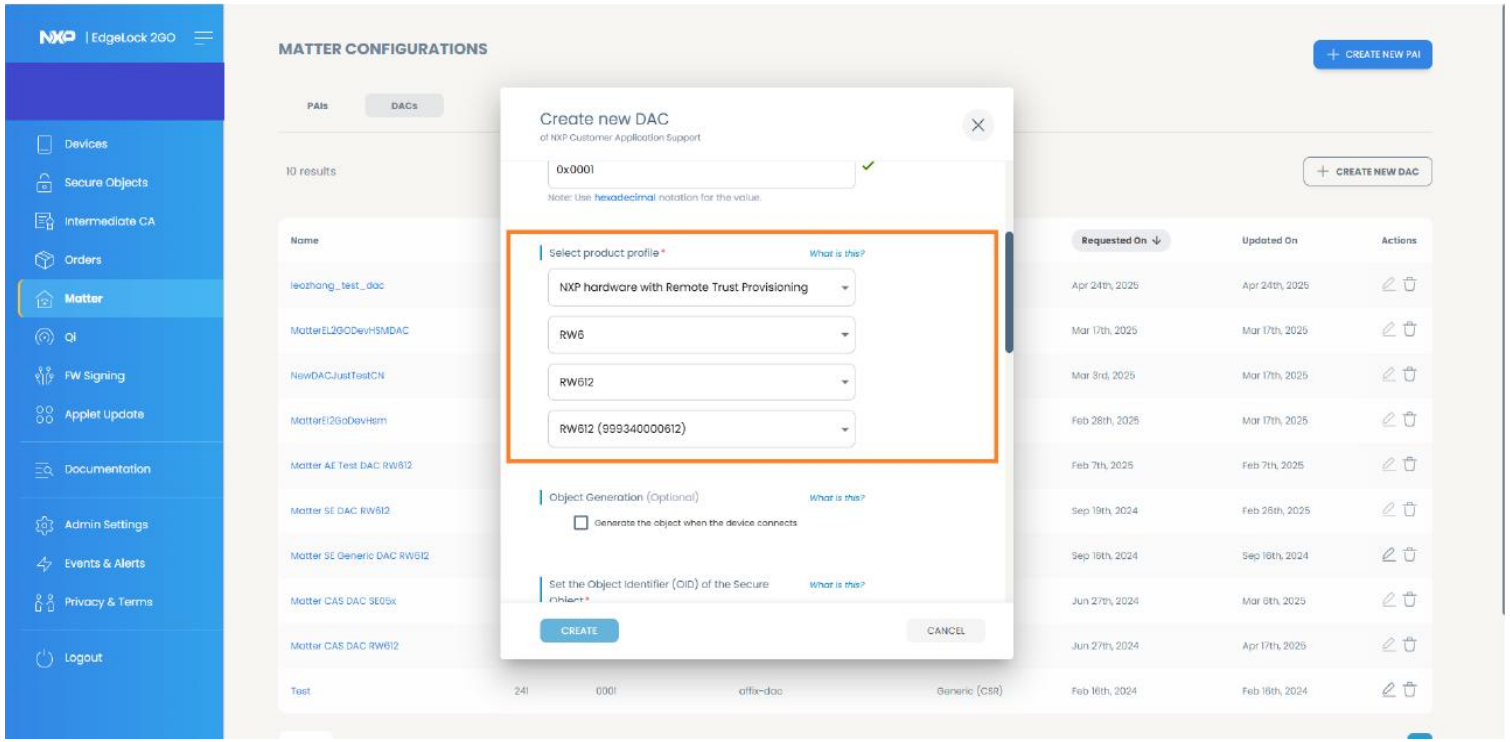
Notes and sources

NXP Public

Matter Manufacturing Flow with EdgeLock2GO

Step 2 - Generate Matter certificates (PAI/DAC/etc.)

- Generate PAI certificate and key
- Create a DAC template
- Create a device group for your devices
- Assign the DAC template to the device group



Notes and sources

NXP Public

Matter Manufacturing Flow with EdgeLock2GO

Step 3 – Retrieve Matter certificates (EL2GO blob)

Use **NXP Secure Provisioning SDK** and the provided el2go application to connect to EdgeLock2GO server and retrieve the secure objects generated for your specific Matter device.

Step 4 - Generate provisioning data (factory data binary)

NXP factory data binary includes:

- DAC & DAC key
- PAI certificate
- SPAKE2+ values (passcode, salt, iteration count, verifier)
- Discriminator
- Serial number (if applicable)

Generated using:
generate_nxp_chip_factory_bin.py

For the EdgeLock2GO flow, use parameters:
--EL2GO_bin
--EL2GO_DAC_KEY_ID
--EL2GO_DAC_CERT_ID

Notes and sources

NXP Public

Matter Manufacturing Flow with EdgeLock2GO

Step 5 – Build your Matter application to use the factory data

Use the **gn** command with:

- **nxp_use_factory_data=true**
- **nxp_enable_secure_EL2GO_factory_data=true**



Step 6 - Flash factory data to designated partition

Write the generated factory data binary to the dedicated flash location: *__FACTORY_DATA_START*

Notes and sources

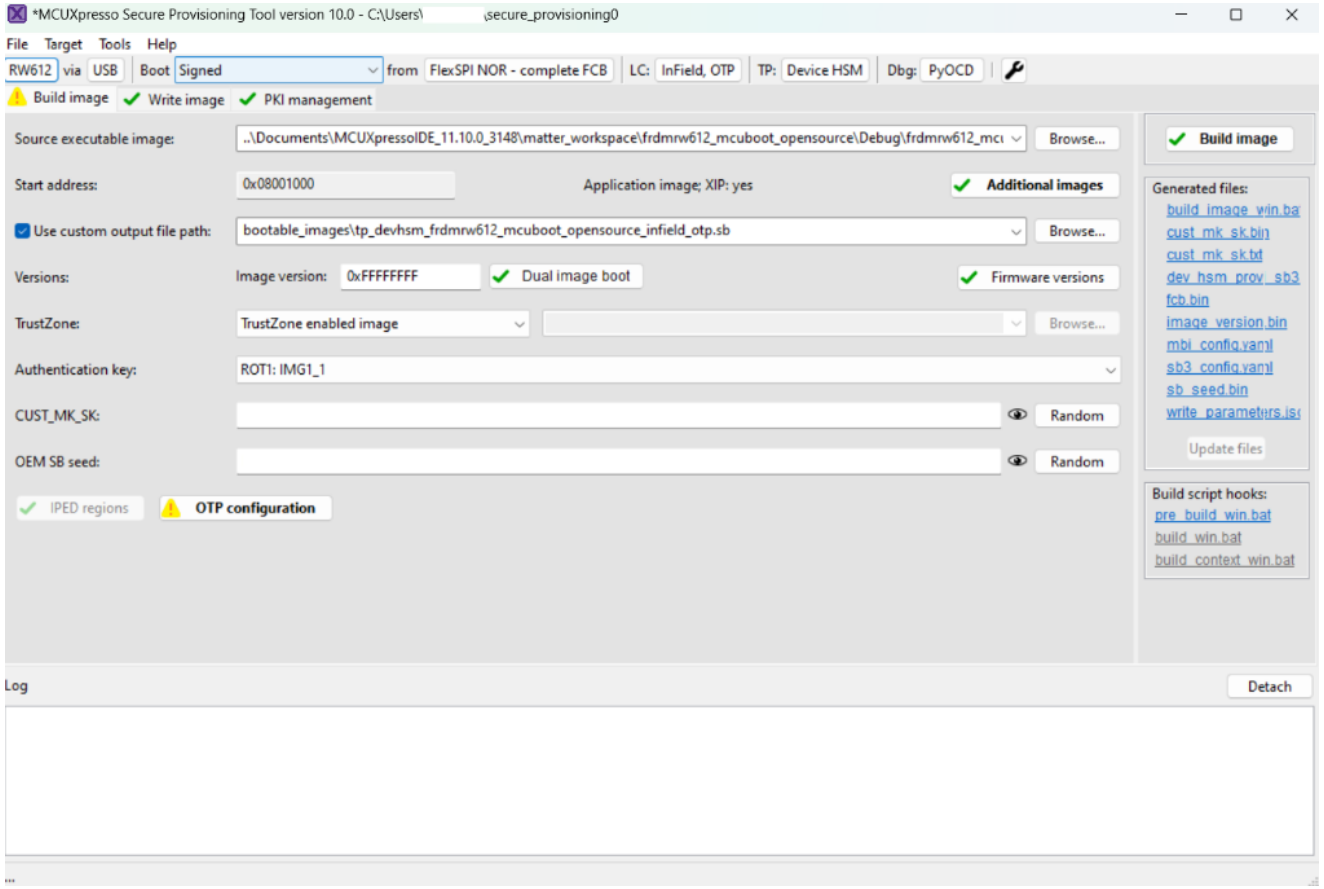
NXP Public

Matter Manufacturing Flow with EdgeLock2GO

Step 7 – Enable secure boot and flash your board

When building your Matter application, add **no_mcuboot=false** parameter and also build a secure bootloader image.

Use MCUXpresso Secure Provisioning Tool to enable secure boot, write your images and manage board lifecycle.

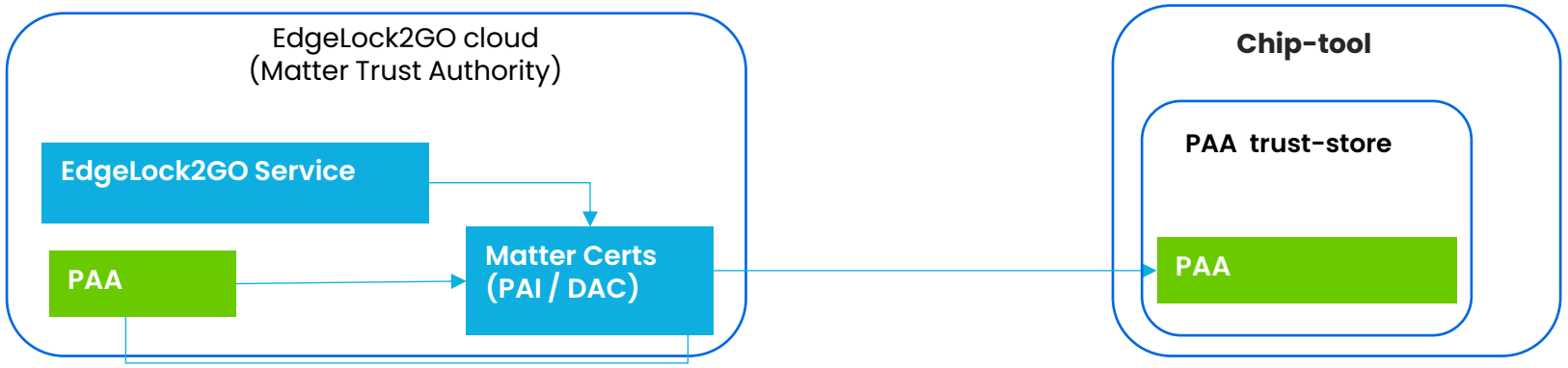


Notes and sources

NXP Public

Chip-tool configuration

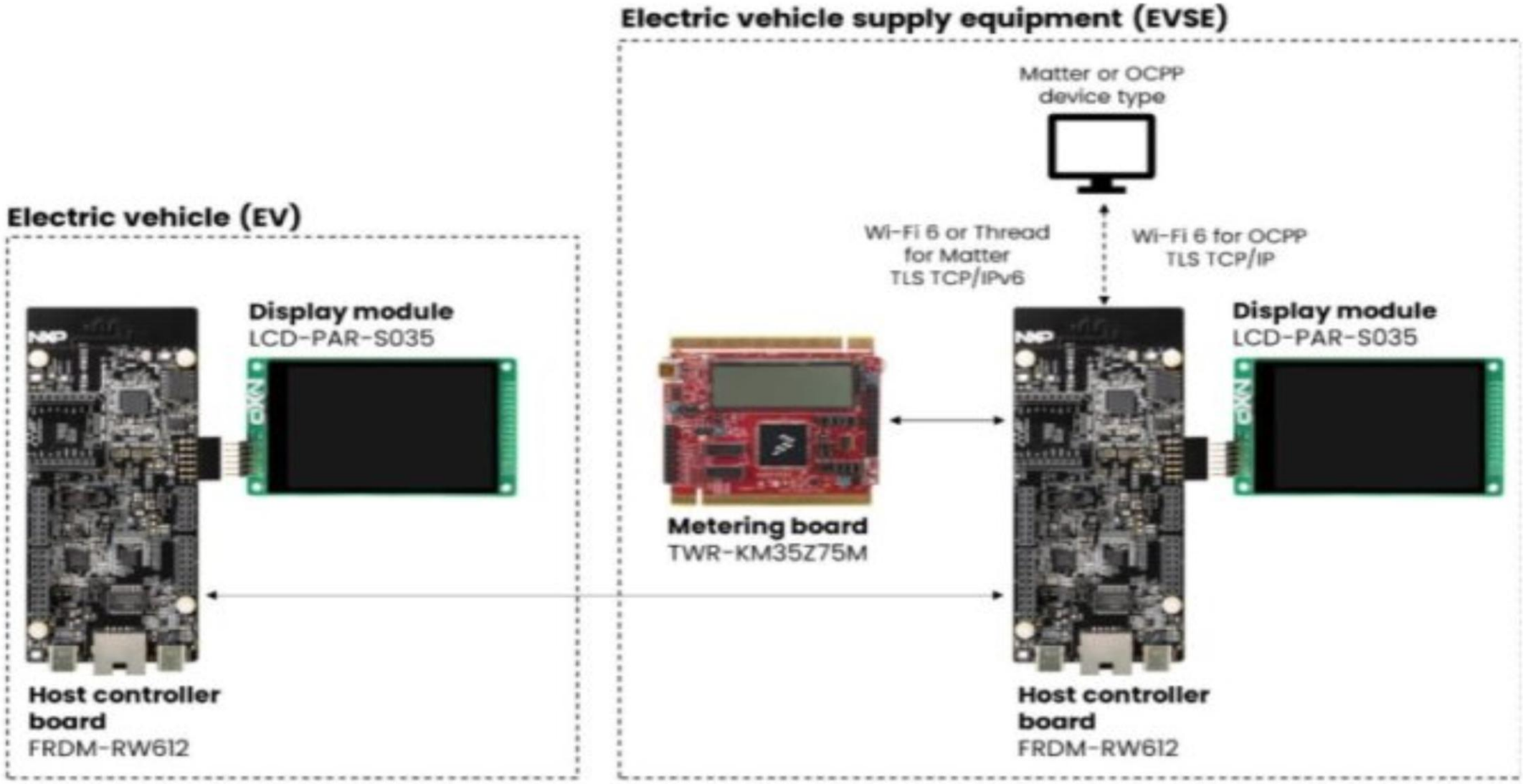
Download the PAA cert in the chip-tool trust store




```
root@imx93evk-iwxxx-matter:~# chip-tool pairing ble-wifi <node_id>
  <wifi_ssid> <wifi_password> 14014 2560 --paa-trust-store-path
 <path_to_el2go_paa_certificate>
```


NXP Public

Demonstration Video



NXP Public

17

Q&A

Get in touch

Susanne Zach, Hitesh Mathukiya, Sri Anand Pochiraju, Sava Alexandru

susanne.zach@nxp.com
hitesh.mathukiya@nxp.com
srianandseshakrishna.pochiraju@nxp.com
alexandru.sava@nxp.com

nxp.com

| **Public** | NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2024 NXP B.V.