

# Getting started with K32W0x1 in Matter





# CONTENTS

1	INTR	RODUCTION								
2	HAR	DWARE SETUP	4							
3	МАТ	TER ENVIRON MENT SETUP	5							
3.1		WSL UBUNTU 20.04 LTS	5							
	3.2	LINUX VIRTUAL MACHINE	6							
	3.3	Setting up Matter Environment	8							
4	K32\	W0X1 MATTER EXAMPLES	11							
	4.1	MATTER APPLICATION BUILDING INSTRUCTION	11							
	4.2	SECOND STAGE BOOTLOADER APPLICATION BUILD.	13							
	4.3	WRITING MATTER APPLICATION TO THE DK6-K32W0x1 BOARD	14							
	4.4	K32W0x1 MATTER APPLICATION DEBUG	18							
	4.5	K32W0x1 MATTER APPLICATION USER INTERFACE	23							
5	MAT	TER NETWORK – CHIP TOOL COMISSIONGING AND CONTROL	25							
6	K32\	W0X1 - EXPLORING WITH MATTER	28							
	6.1	ADDING MANUFACTURING DATA TO THE APPLICATION	28							
	6.2	OVER THE AIR UPGRADE – FLASH CONFIGURATION	32							
	6.3	Reading Tokenizer Logs	32							
	6.4	ZAP TOOL. UPDATE CLUSTER/ENDPOINT FUNCTIONALITY	33							



### **1** INTRODUCTION

**Matter** (previously known as Project CHIP) is a new single, unified, application-layer connectivity standard designed to enable developers to connect and build reliable, secure IoT ecosystems and increase compatibility among Smart Home and Building devices.

For enabling Matter devices, NXP offers scalable, flexible and secure platforms to enable the variety of use cases Matter addresses – from end nodes to gateways – so device manufacturers can focus on product innovation and accelerating time to market.

This document focuses on NXP's standalone solution for Matter end nodes/routers with Thread using the K32W0x1 wireless microcontroller family.

The K32W0x1 portfolio is designed for ultra-low-current multiprotocol wireless IoT devices with support for IEEE 802.15.4 mesh network protocols Zigbee<sup>®</sup> and Thread<sup>™</sup> as well as Bluetooth<sup>®</sup> Low Energy 5.0. These wireless MCUs include multiple low-power modes and ultra-low radio Tx and Rx power consumption which enables IoT products powered by K32W0x1 to have extended battery life. With high Rx sensitivity and configurable Tx output power, the K32W0x1 MCUs offer reliable and robust connectivity performance.

https://www.nxp.com/part/K32W061#/ https://www.nxp.com/part/K32W041#/



### 2 HARDWARE SETUP

The minimum hardware required to create and run an end to end Matter setup with K32W0x1 is listed below:

- i.MX8M Mini EVK acting as a Thread Border Router and Chip Controller: 8MMINILPD4-EVK
- USB-K32W0x1 acting as RCP for Thread Network: USB-K32W0X1
- DK6-K32W061 + Expansion board OM15082- acting as Matter Accessory Device: IOTZTB-DK006
- SE051H Secure Element- optional, to be used for cryptographic operations: <u>SE051</u>





### **3 MATTER ENVIRONMENT SETUP**

<u>Matter</u> development relies on open-source resources, leveraging Linux based operating systems like Ubuntu and other tools like git, gcc and python. This also includes <u>GN</u>, a meta build system that generates makefiles and <u>Ninja</u>, a build system meant to replace Make tool.

Matter is available as an open-source SDK containing all the necessary components from scripts to install required tools to stack source code and vendor provided applications.

First step in developing a Matter application is to have Linux support for the build. The recommendation is to have a native Linux machine. If Windows is preferred operating system, support for the build can be set by using:

- Windows Subsystem for Linux (WSL);
- Linux Virtual Machine;

## 3.1 WSL Ubuntu 20.04 LTS

The Windows Subsystem for Linux (WSL) provides a GNU/Linux environment directly to Windows.

Use the following steps to install the WSL Ubunut 20.04 LTS:

1. On Windows 10, open PowerShell as administrator and run the following commands:

Enable the Windows Subsystem for Linux:

dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

Enable virtual machine feature:

dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart

Restart your machine to complete the WSL installation

2. Install the Ubuntu 20.04 from Microsoft Store



Create a user account and password for your new Linux distribution



After installation is complete, run Ubuntu 20.04 LTS as administrator. Administrator rights are required on the first run to install needed packages for Matter.

All Apps Documents Setting	gs Email I	More ▼
Best match		
O Ubuntu 20.04 LTS App	$\rightarrow$	Ø
Apps           O         Ubuntu 18.04 LTS           Documents - This PC (4+)	>	Ubuntu 20.04 LTS <sub>App</sub>
Folders (2+)		□ <sup>-</sup> Open
		Run as administrator
		-🏳 Pin to Start
		-🛱 Pin to taskbar
		Ø App settings
		☆ Rate and review
		🔄 Share
		🔟 Uninstall
𝒫 ubuntu 20.04 LTS		

## 3.2 Linux Virtual Machine

The following steps guide you through virtualbox machine installation steps:

- Download and install virtual machine: https://www.virtualbox.org/wiki/Downloads
- Please consider that based on your operating system some extra steps will be required for enabling the Virtualization support. These extra steps consist in settings that need to be done in the machine's BIOS regarding virtualization support and they depend on the machine type and manufacturer. For further details, please consult your machine's user manual.
- Download the <u>desktop image for Ubuntu 20.04 Focal</u> and create virtual machine using the downloaded ISO.
- Configure the virtual machine:
  - Linux Ubuntu 20.04 (64 bit);
  - VM disk size more than 20GB;



- Enable USB controller -> USB 1.1(OHCI) Controller (support for USB 2.0 is recommended, if possible, this being available via the <u>Oracle VM VirtualBox Extension Pack</u>);
- Enable network adapter-> Adapter 1 -> Attached to Bridged Adapter;
- Other recommended settings are to increase the number of cores and the size of RAM to be dedicated to the virtual machine. This depends on the user's actual machine. Minimum recommendation is to have 4 dedicated cores and 4 GB of RAM.

			· · ·
💱 Oracle VM VirtualBox Manager			- 🗆 ×
File Machine Help			
Tools	New Settings Discard Start		
Ubuntu 20.04	General Name: Usuntu 20,04 Oceanor System: Ubuntu (54-bb)		Preview
	🥹 Ubuntu 20.04 - Settings	? ×	
	General		Ubuntu 20.04
	System Basic Advanced Description Disk Encryption		
	Display Name: Ubuntu 20.04		
	Type: Linux	- 64	
	Version: Ubuntu (64-bit)	· ·	
	4 Audio		
	Network		
	Serial Ports		
	USB USB		
	Shared Folders		
	User Interface		
		DK Cancel	



### 3.3 Setting up Matter Environment

The following steps guide you through creating Matter build environment for K32W0x1 MCU.

1. Matter Dependencies:

Check for updates and install dos2unix (useful for WSL):

- \$ sudo apt update
- \$ sudo apt upgrade --y

*\$ sudo apt-get install dos2unix # optional, required if using files having DOS style line endings that need to be converted to Linux style line endings* 

Install Matter dependencies:

\$ sudo apt-get install git gcc g++ python 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 gcc-arm-none-eabi

~\$ sudo apt-get install git gcc g++ python 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
Reading package lists Done
Building dependency tree
Reading state information Done
Note, selecting 'python-is-python2' instead of 'python'
g++ is already the newest version (4:9.3.0-1ubuntu2).
gcc is already the newest version (4:9.3.0-1ubuntu2).
libcairo2-dev is already the newest version (1.16.0-4ubuntu1).
pkg-config is already the newest version (0.29.1-0ubuntu4).
python3-dev is already the newest version (3.8.2-0ubuntu2).
unzip is already the newest version (6.0-25ubuntu1).
ninja-build is already the newest version (1.10.0-1build1).
python-is-python2 is already the newest version (2.7.17-4).
python3-venv is already the newest version (3.8.2-0ubuntu2).
git is already the newest version (1:2.25.1-1ubuntu3.5).
libavahi-client-dev is already the newest version (0.7-4ubuntu7.1).
libdbus-1-dev is already the newest version (1.12.16-2ubuntu2.2).
libgirepository1.0-dev is already the newest version (1.64.1-1~ubuntu20.04.1).
libglib2.0-dev is already the newest version (2.64.6-1~ubuntu20.04.4).
libssl-dev is already the newest version (1.1.1f-1ubuntu2.16).
python3-pip is already the newest version (20.0.2-5ubuntu1.6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
~\$
~\$

Restart the Linux machine/environment if required.

2. Getting the K32W0x1SDK:

The <u>MCUXpresso SDK builder website</u> allows the user to create SDKs for K32W0x1 by choosing the option for Host OS to be Linux and selecting all middle ware components. For the following steps used as example, the SDK 2.6.8 for the K32W061 variant was used. Download and unzip into a predefined location on your PC (for example on Windows OS, C:/nxp or on Linux based OS, /home/user/Documents).

10/2 ·····			settim-reserroororood er-oo	8		រេ
MCUXpresso S	SDK Builde	er 🔹 • Internal 📫 Create :	lira ticket			0 • E
	Build Generate Developer Selections he	A downloadable SDK archive for use v Environment Settings re operaning hinst system, boothain or middleware Host OS	DK6 with desktop MCUXpresso Too	ols. a redukted in the SDK and Generated Projects Toxebrain / DE R O De Tox	SDK Version SDK Tag	2.6.8 (released 2022-07-17 REL_SDK_JN_QN_K32W,
ADMINISTRATION						
Notifications	Search.			Q SELE	CT ALL	UNSELECT ALL
Company carps		Name	Category	Description	Depen	dencies
MCUXpresso IDE		AWS IoT Core	Middleware	Amazon Web Service (AWS) IoT Core SDK		
MCUXpresso		CMSIS DSP Library		CMSIS DSP Software Library		
Offine data		NTAG I2C	Middleware	NTAG I2C (plus) communication library		
MCUXpresso		Wireless BLE stack	Middleware	BLE		
INTERNAL	82	Wireless OpenThread stack	Middleware	OpenThread		
1 Deployed Releases		Wireless zipbee stack	Middleware	ZIGBEE		
<ul> <li>Hardware in Releases</li> <li>브 Analytics</li> </ul>		FreeBTOS		Basilima operation system for microcontrollers from	Amazon	

If the downloaded SDK is not the Linux host variant, then the following command must be run to resolve errors with the file formatting for the SDK intended for a non-Unix host system:

\$ find . -type f -print0 | xargs -0 dos2unix

3. Matter Building Setup instructions:

Clone the Matter SDK using the public repo:

\$ git clone https://github.com/NXPmicro/matter/ \$ cd matter \$ git checkout v1.0-branch-nxp \$ git pull \$ git submodule update --init -recursive

Start build environment by running the activate script: \$ source ./scripts/activate.sh

Note that the *activate.sh* script can take a long time to execute; this behavior is normal.



patching file fsl\_os\_abstraction\_free\_rtos.c Hunk #3 succeeded at 473 with fuzz 1. Hunk #3 succeeded at 524 with fuzz 2. patching file ofautils.h K32W SDK 2.6.6 was patched! -/natterS -/natte

/matter\$



# 4 K32W0X1 MATTER EXAMPLES

In the current Matter SDK for K32W0x1 platform we are providing reference examples for end nodes, Thread End Device type applications, with support for low power, over the air update and cluster configuration and control.

The examples are listed in the matter-> examples:



These applications provide support for persistence data storage (PDM) used for storing Matter and Thread configuration data. These are stored in the external flash memory available on the DK6 board as Macronix MX25R8035F connected through Quad SPIFI interface or the stacked flash of the same variant in the K32W041AM part connected through Dual SPIFI interface. The available space on this memory is 8 Mbit (1 MB). The SPIFI interface on the K32W0x1 part supports up to 16 MB of external flash space.

## 4.1 Matter application building instruction

The following build steps are based on the lighting app reference for NXP K32W0X1 - Building steps: <u>https://github.com/NXPmicro/matter/tree/v1.0-branch-nxp/examples/lighting-app/nxp/k32w/k32w0</u>

- Set the environment variable NXP\_K32W0\_SDK\_ROOT to where you extract the SDK. An example is shown below, where the SDK is unzipped into the ~/Documents/SDK\_2\_6\_8\_K32W061DK6 directory:
   \$ export NXP\_K32W0\_SDK\_ROOT=~/Documents/SDK\_2\_6\_8\_K32W061DK6
  - Apply SDK patch from the Matter root directory using the following command:
- \$./third\_party/nxp/k32w0\_sdk/sdk\_fixes/patch\_k32w\_sdk.sh
  - Build K32W0x1 example application. Adapt the gn command with the device configuration by enabling/disabling the compile options:
    - Chip type (061/041/041A/041AM) for example, for K32W041AM, build\_for\_k32w041am
    - OTA requestor feature: *chip\_enable\_ota\_requestor (true by default)*
    - Low power support: *chip\_with\_low\_power* (available for contact sensor and lock apps)
    - Secure element SE015H usage: chip\_with\_se05x
    - OM15082 Expansion board attached to DK6 board: chip\_with\_OM15082
    - OpenThread command line interface: *chip\_with\_ot\_cli*
    - Manufacturing data like DAC and PAI certificates and other security related data: chip\_with\_factory\_data
    - Support for pigweed tokenizer applied on the constant strings: chip\_pw\_tokenizer\_logging
    - Support for OpenThread CLI: *chip\_with\_ot\_cli*
    - In case the K32W0x1 chip is used on a board without a 32KHz crystal, there is the option of using the internal 32KHz free running oscillator as a clock source. This can be enabled by setting use\_fro\_32k=1



\$ cd examples/lighting-app/nxp/k32w/k32w0 \$ gn gen out/debug --args="k32w0\_sdk\_root=\"\${NXP\_K32W0\_SDK\_ROOT}\" chip\_with\_OM15082=1 chip\_with\_ot\_cli=0 is\_debug=false chip\_crypto=\"tinycrypt\" chip\_with\_se05x=0 chip\_pw\_tokenizer\_logging=true mbedtls\_repo=\"//third\_party/connectedhomeip/third\_party/nxp/libs/mbedtls\""

• Compile the demo application with the following command: \$ ninja -C out/debug

• Sign the image:

\$ \$NXP\_K32W0\_SDK\_ROOT/tools/imagetool/sign\_images.sh out/debug/

• If permission denied error is encountered, run the following command: \$ chmod +x \$NXP\_K32W0\_SDK\_ROOT/tools/imagetool/sign\_images.sh

• There might be also a Python error related to pycrypto which can be fixed by issuing: \$ pip3 install pycrypto \$ pip3 install pycryptodome

Recommended versions for these Python packages are: \$ pip3 list | grep - i pycrypto pycrypto 2.6.1 pycryptodome 3.9.8





- After build is complete, the results can be found in *examples/<application\_name>/nxp/k32w/k32w0/out/debug* folder, which contains the elf and bin files for the application.
- For example, for the lighting app, in the examples/lighting-app/nxp/k32w/k32w0 folder, user can find chip-k32w0x-light-example as elf file and chip-k32w0x-light-example.bin as binary file.

# 4.2 Second Stage Bootloader application build

Using the *chip\_enable\_ota\_requestor* build option in the Matter application build will automatically enable overthe-air update support. This means that the user must program two application binaries into the board, the first being the Second Stage BootLoader (SSBL) and the second being the binary obtained in the Matter application Building instructions step.

If *chip\_enable\_ota\_requestor* is set to false, then only the Matter application binary is required; however, the device will be unable to receive over-the-air firmware updates.

Below are are steps to build the SSBL application:

- As a prerequisite, the user must have the MCUXpresso IDE installed for this step, and the K32W0x1 SDK imported in the IDE.
- The SSBL can be built from the SDK demo examples:

Import SDK example(s) -> select wireless->framework->ssbl application.

Import projects	
9	
oject name prefix: k32w061dk6	2 Project name suffix:
Use default location	
cation: /home/user/Documents/MCUXpresso_11.0.1_2563/workspace/	k32w061dk6 Brows
roject Type	Project Options
○ C Project ○ C++ Project ○ C Static Library ○ C++ Static Library	SDK Debug Console  Semihost UART  Example default Copy sources Import other files
xamples	🔟 🖉 🙀 🖲
type to filter	
ame Descript	ion Version application

- The SSBL project must be compiled with the PDM\_EXT\_FLASH define:





 Next step is to build the project using MCUXpresso. The resulted binary file will be located in the corresponding debug/release folder: k32w061dk6\_ssbl.bin

# 4.3 Writing Matter application to the DK6-K32W0x1 board

DK6 Hardware connections:



The Matter application can be downloaded into the DK6 K32W061 board either:

- via the LPC-LINK2 USB port using an IDE debugger (for example, MCUXpresso, Jlink);
- via UARTO using the DK6 Flash Programmer;



Flashing the application using DK6Programmer:

- Connect the DK6 board using a mini-USB cable to the connector marked with FTDI USB.
- Set the J4 and J7 jumpers to the middle position (JN UARTO FTDI)



- The DK6 Programmer installer is included in the {*SDK path*} -> tools folder:

<SDK\_root>\tools\JN-SW-4407-DK6-Flash-Programmer

- Once the application is installed, the COM port for K32W061DK6 can be identified by running the following command from a powershell terminal or command prompt:
- DK6Programmer.exe --list
  - Use the following commands to write to the board:
    - To erase the internal flash:

DK6Programmer.exe -V 5 -P 1000000 -s <COM\_PORT> -e Flash

To write the SSBL at address 0x0:

DK6Programmer.exe -V2 -s <COM\_PORT> -P 1000000 -Y -p FLASH@0x00="k32w061dk6\_ssbl.bin"

- To write the PSECT containing the image directory:
- First, image directory 0 must be written:

DK6Programmer.exe -V5 -s <COM\_PORT> -P 1000000 -w image\_dir\_0=000000010000000

Where the interpretation of the image directory 0 fields is:00000000-> start address 0x000000001000-> size = 0x0010 pages of 512-bytes (= 8kB)00-> not bootable (only used by the SSBL to support SSBL update)

00 -> SSBL Image Type

- Second, image directory 1 must be written:

DK6Programmer.exe -V5 -s <COM\_PORT> -P 1000000 -w image\_dir\_1=00400000C9040101



Where the interpretation of the image directory 1 fields is:00400000-> start address 0x00004000CD04-> 0x4C9 pages of 512-bytes (= 612,5kB)01-> bootable flag01-> image type for the application

- To write the application at address 0x4000:

DK6Programmer.exe -V2 -s <COM\_PORT> -P 1000000 -Y -p FLASH@0x4000="chip-k32w0x-light-example.bin"

#### Flashing the application using LPC-Link2 – J-Link Firmware:

By default, the LPC-Link2 is configured with CMSIS-DAP firmware. The J-Link firmware is also supported by this debugger and the following instructions can be used to change between the firmware versions.

- Put the board in DFU mode for debugger (JP5 – DFU mode) and connect the board to the PC via the LPC-LINK2 USB



- Download LPC Script according to the host OS (Windows, Linux, etc.) and install.

#### For Windows:

- Go to the installation folder (for example, by default, C:\nxp\LPCScrypt\_2.1.2\_57\)
- In the \scripts folder, open a Command Prompt and run the program\_JLINK.cmd batch file.
- Make sure that the board is connected and press any key to continue.
- Unplug the board and switch the JP5 back to NORMAL.
- The firmware should be flashed to the LPC-Link2 LPC chip.

#### For Linux

- Go to the download folder and install the LPC script: \$ chmod +x lpcscrypt-2.1.2\_57.x86\_64.deb.bin



\$ ./lpcscrypt-2.1.2\_57.x86\_64.deb.bin

- Once installed, execute the script to reprogram your board with JLink firmware:
  - \$ <LPCScrypt\_InstallDir>/scripts/program\_JLINK
- Unplug the board and switch the JP5 back to NORMAL.
- The firmware should be flashed to the LPC-Link2 LPC chip.
- To revert to the CMSIS-DAP firmware, the program\_CMSIS script needs to be run from the same folder using the same procedure



*Note: To revert to the CMSIS-DAP firmware, the program\_CMSIS script needs to be run from the same folder using the same procedure.* 

- Download and install <u>J-Link</u> on your computer;
- Connect the board to the PC;
- Launch J-Link command to connect to the device from the installation folder:
- From Windows:
  - In C:\Program Files (x86)\SEGGER\Jlink folder, open a Command Prompt and run:

\$ Jlink.exe -device K32W061 - if SWD - speed 4000 - autoconnect 1

- From Linux

\$ JLinkExe -device K32W061 -if SWD -speed 4000 -autoconnect 1

- Erase the image using *erase* command
- \$ J-Link>erase
  - Download new image using *loadbin* command (please consider that for example with SSBL, the start address will be 0x4000, otherwise it is 0)

\$ J-Link>loadbin <path/to/binary/ k32w061dk6\_ssbl.bin, 0x0000



#### *\$ J-Link>loadbin <path/to/binary/chip-k32w0x-light-example.bin, 0x4000*

Note: Please consider that the PSECT configuration using DK6Programmer will be also required first time.



## 4.4 K32W0x1 Matter Application Debug

The following steps can be applied for Windows as well as for Ubuntu host OS:

- A debugger must be used here, whether it's the onboard LPC-Link2 debugger connected through the DK6 board's LPC-LINK2 USB port, or an external debugger connected to the JTAG on the same board.
- Download and install the <u>MCUXpresso IDE</u> (version >= 11.0.0) and create a workspace.
- Import K32W0x1 SDK into the IDE. This can be done by drag-and-drop the SDK archive into the *MCUXpresso IDEs Installed SDKs* tab.
- Import *connectedhomeip* repoin MCUXpresso IDE as Makefile Project. Use none as Toolchain for indexer settings: *File -> Import -> C/C++ -> Existing Code as Makefile Project*



		Create a new makenie project in a directory containing existing code	_
		Select as import uitset	
		face filer test	
		v in General	
		D Archive File	
🗶 workspace - MCUIgresse IDE	- 0 ×	2 Existing Projects into Workspace	
File Edit Navigate Search Project ConfigNosis Run MIDS Analysis Windo		System	
Print Lation I. Effection & Late V. Brishenky		import projects(s) from XML description	
8 8 7   8 5   8 +	1 (1 (0 100) (1 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0 (0	<ul> <li>Preferences</li> <li>Projects from Echler or Archive</li> </ul>	
There are no projects in your workspace. To add a project		- D C/C++	
Course a new MC/Revenue IDE C/C++ project.		C/C++ Executable	
<ul> <li>Import exemples from SDK.</li> </ul>		di CiCo divini Serving	
is most promit.		III) Existing Code as Makenie Project	
		> Device Configuration Tool	
		> 🗁 Git	
		> install	
		> In MCUXpresso Config Tools	
		Dian Team	
		> as XML	
		(7) < Back Next > Finish Cancel :	
	Strateled UDs 1: Direpeters & Institutes #Terminal Dicensite & Integration & Menory @Delogoe Conside In-Hosp and Stadk Upger 4/Seach 3; Office Respireds EVAD Text Config		
	ip installed SDRs		
OrQuickstart Rowill III III Variables % Beakpoints -	Do mate as 300, umps pag and drop as 300 tag findhodic into the financial (Diri view (Common Incogness) finder) (Incole 500), Analtate Science, Nanciae Science, Common Incogness) finder)	New Project	×
MCUXpresso IDE - Quickstart Panel		-	
No period adarbati			
CBC No project advocad + Create as impact a project	124 002,2,0000000 244 350 B	Import Existing Code	
CRC No project whether In Counter an impart to project These project.	244 330 \$ (mm- 92033000000)	Import Existing Code	
CRC To project advected	2490,0,00000 [244 149 # (90,0,0,000,00.ag	Import Existing Code Create a new Makefile project from existing code in that same directory	
CRC To projekt wincht	11 (10)(10000 ) 111 11 J 90(10000000	Import Existing Code Create a new Makefile project from existing code in that same directory	
Construction of the specific Construction of the specific of the specific Construction of the specific of	10/00/00/000 144 14 J 00/00/00/049	Import Existing Code Create a new Makeflie project from existing code in that same directory Project Name	
Torm to proper units of the second	11 (10)(10)(10)(10) (10) (10) (10)(10)(10)(10)(10)(10)(10)(10)(10)(10)	Import Edisting Code Orasis a new Makefile project from existing code in that same directory Project Name Imater Imater	
The target stands		Import Existing Code Create a new Makefile project from existing code in that same directory Project Name matter	
The topped sense         Control topped sense           Control topped sense         Control topped sense		Import Existing Code Create a new Makellie project from existing code in that same directory Project Name [matter Existing Code Location	
The topped name       Colorant lange stages		Import Exiting Code Create a new Makelle project from existing code in that same directory Project Name matter Existing Code Location C.nutter Bit	rowse
With the page a matter           Content is a matter is is		Import Existing Code Create a new Makellie project from existing code in that same directory Project Name (matter Existing Code Location [C.matter ] [Bit	rowse_
The Target senses Colors Calest arget ( The Target Sense ■ arget particular to a sense ■		Import Exiting Code Create a new Makellie project from existing code in that same directory Project Name Imster Existing Code Location C.t.matter Languages	rowse_
The Name and State		Import Existing Code Create a new Makellie project from existing code in that same directory Project Name Imater Existing Code Location C.n.mater Languages Exit Code	rowse_
Min     Name       Image: State Sta		Import Existing Code Great a new Makelle project from existing code in that same directory Project Name Imate Existing Code Location C_nmatter Existing Code Location C_nmatter C_nmatter C_C_C_C++ Torcheals to Indexer Settings	rowse
© Spranger Spr		Import Existing Code Create a new Makelle project from existing code in that same directory Project Name Imster Existing Code Location C.nutter Logguages ⊡ C G C ++ Tookhain for Indexer Settings	rowse
CON Experiment Constraints C	1000/000000000000000000000000000000000	Import Existing Code Greate a new Makellie project from existing code in that same directory Project Name matter Existing Code (acation Committer Existing Code (acation	rowse_
CR: Spranger S		Import Existing Code           Create a new Makellie project from existing code in that same directory           Project Name           matter           Institute Code Location           Currate           Languages           2C (2C++)           Toolchain for Indexed Settings           Expansion           Cognition Constraintion	rowse
Constraints     Constrai	1000/000000000000000000000000000000000	Import Existing Code Create a new Makellie project from existing code in that same directory Project Name Imate Existing Code Location Crumter Languages C C C C +  Toolchain for Indees Settings Tool C C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C Spain Code Not P Act C +  C C C +  C C +  C C +  C C +  C C +  C C +  C C +  C C +  C C +  C C +  C C +  C +	rowse_
COS Sprate Section 2015 Cost S		Import Exiting Code           Create a new Makellie project from existing code in that same directory           Project Name           mater           Exiting Code Location           Clumate           Clumate           Exit Cgle+           Toolchain for Indexer Settings           Copen GCC           (Option GCC)           (Option GCC)	rowse
© September 2015 September 2		Import Existing Code Create a new Makelle project from existing code in that same directory Project Name Institer Existing Code Location Committer Existing Code Location Committer Committee Settings Common Committee Settings Committe	rowse_
CR: Sprander CR: Sprander Sprander CR: Sprander CR: Sprander		Import Existing Code Create a new Makellie project from existing code in that same directory Project Name matter Existing Code Location C-Limatter Existing Code Location C-Limatter Existing Code Location C-Limatter Code Location C-Limatter Code Location C-Limatter Code Location C-Limatter	rowse
CR: Spranger S		Import Existing Code Create a new Makelle project from existing code in that same directory Project Name matter Existing Code Location C.c.matter Location C.c.matter Toolcalin for Indexe Settings Toolcalin for Indexe Settings Code Settings	rowse_
CR: Sprane with the second se	1 (19) (1,10000 (1)) 1 (1) (1) (1) (1) (1) (1) (1) (1) (1) (1	Import Existing Code           Great a new Makellie project from existing code in that same directory           Project Name           matter           Existing Code Location           CL,matter           Existing Code Location           CL,matter           Existing Code Location           CL,matter           Existing Code Location           CC, C, C + +           Tochchain for Indeer Settings           Copen of C           (RNU Foots           Protochain NoP MCU Toots	rowse_
CR: Reproduction (C)		Import Existing Code           Create a new Makellie project from existing code in that same directory           Project Name           matter           Institute Code Location           Currate           Languages           CC (Currate           Code (C) (C++)           Toolchain for Indexer Settings           Propert GoCo.           Code (C) (C++)           Code (C) (C++)           Code (C) (C++)           Code (C++)	rowse
© Representation of the second of the secon		Import Existing Code           Create a new Makelle project from existing code in that same directory           Project Name matter           Existing Code Location           Contraine           Comparison           Contraine           Existing Code Location           Comparison           Contraine           Comparison           Contraine           Contraine           Copymen GCC           City Automotic Toocham           Not MCU Tools	rowse
CR: Representation of the second of the sec		Import Exiting Code           Create a new Makellie project from existing code in that same directory           Project Name           mater           Exiting Code Location           Cmatter           Earguages           CC. (Catter           CC. (Catter           Concest           Concest     <	rowse
CR: Spranger S		Import Existing Code           Create a new Makellie project from existing code in that same directory           Project Name matter           Existing Code Location           C.nutler           C.quarter           Existing Code Location           Code and the Code of	rowse
CR: Representation of the second of the sec		Import Exiting Code       Create a new Makellie project from existing code in that same directory       Project Name       mater       Exiting Code Location       C.c.mater       Languages       Exit C C + +       Column for Indexe Settings       Conventse       Column for Convertse       Convertse    <	rowse_

Select

×□

- Configure MCU Settings:
   Right click on the Project -> Properties -> C/C++ Build -> MCU Settings -> Select K32W061 -> Apply & Close
- Configure the toolchain editor: Right click on the Project -> C/C++ Build-> Tool Chain Editor Uncheck "Display compatible toolchains only" and select NXP MCU Tools from "Current toolchain" drop box.

Apply & Close

Properties for matter		U	× Properties for matter	U
type filter text	MCU settings	<b>⇔</b> • <	* 1 Tool Chain Editor	
yelped ton text     yelped to text     yelped tot     yelped to text     yelped tot     yelped	MCU estings Available parts - OC MON -		Construction     C	1 → Manage Configuration solution alter
	Memory details (K32W061)			
	Default Liefderer Final Deter Type Nemer Allen Fisch PROCEAULS, Fisch Ram Soland J. Aus Ram Soland I. Au			Retries Default
0		Apply and Close Cancel	0	Apply and Close Cancel

- Create a debug configuration:



*Right click on the Project -> Debug As->MCUXpresso IDE LinkServer (inc. CMSIS-DAP) probes -> OK -> Select elf file* 

Probes discovered Connect to target: K32W00 1 probe found. Select the probe	to use:				Select binary for matter 168 binaries found in matter Select binary to use.			- 0
Available attached pro	bes				Launch Configuration Selection			
Name	Serial number / I_	Туре	Manufacturer	IDE Debug Mode	Name	Image to Debug	Location	Launch Type
LPC-LINK2 CMSIS-DAP V5	CTAYAQIR	LinkSe_	NXP Semicon_	Non-Stop	examples/lighting-app/nxp/k32w/k32w0/out/debug/chip-k32w0x-light-example	2		
					mind_party/ocurraiolab/bioc2_sok/repo/image_cont/nash_build			
					third_party/bouffalolab/bi602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
					third_party/bourtalolab/bib02_sdk/repo/toolchain/riscv/Linux/bin/riscvb4-unknown-e.			
					third_party/bourtaiolab/bi602_sdk/repo/tooichain/riscv/Linux/bin/riscv64-unknown-e			
					third_party/bouffaloiah/bi602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e			
					third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
Supported Prober (tick/untick to	anable/dicable)				third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
Supported Probes (incounties in	(and chick phone)	-			third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
MCUXpresso IDE LinkServer	(Inc. CMSIS-DAP) pr	obes			third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
P&E Micro probes					third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
SEGGER J-Link probes					third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
Probe search options					third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e_			
					third_party/bourtaiolab/bi602_sok/repo/tooichain/riscv/Linux/bin/riscv64-unknown-e			
Search again					third_party/houffalolab/hi602_sdk/repo/toolchain/riscv/Linux/hin/riscv64.uninown-e			
Remember my selection (for t	his Launch configura	tion)			third_party/bouffalolab/bl602_sdk/repo/toolchain/riscv/Linux/bin/riscv64-unknown-e.			
								OK Cancel
(?)			OK	Cancel				
	LINK Mars							

- Set the Connect script for the debug configuration to QN9090connect.scp from the dropdown list: Right click on the Project -> Debug As -> Debug configurations... -> LinkServer Debugger

	Debug Configurations	- 🗆 X
lep 第3. ゆぶばざ 8 0 白 / ノ 1 ゆ・ 5 - 0 - 0 - 0 - 0 - 1 回目 1 日 1 - 5 - 5 - 5 - 7 - 1 1	Create, manage, and run configurations	\$
launch history)	🖸 🖉 🖗 🗎 🗶 🖻 🐎 🔹	Name: matter LinkServer Default
Datug As	[	Main & GDB Debugger LinkServer Debugger GUI Flash Tool Other Symbols > Startup > Source Common
	Concernent and Architecture     Concernent and Architecture     Concernent Archit	ElidiServer Experience     Elidiserver Exp
		Additional options
👌 Installed SDKs 🗇 Properties 🔮 Problems 🛷 Terminal 🔍 Console 🕮 🚔 Image Info 🔞 Memory 🗟 Debugger Console 🚥 F		Pre launch command
IDT Build Console (matter) 15:44:52 **** Build of configuration Default for project matter ****	< >> Filter matched 15 of 15 items	Revert Apply
	1	Debug Close

 Set the Initialization Commands: Right click on the Project -> Debug As -> Debug configurations... -> Startup set non-stop on set pagination off set mi-async set remotetimeout 60000 ##target\_extended\_remote## set mem inaccessible-by-default \${mem.access} mon ondisconnect \${ondisconnect} set arm force-mode thumb \${load}



X Debug Configurations	- 0	×
Create, manage, and run configurations	1	ñ.
Create, manage, and run configurations	Name: matter LinkServer Default           Main (* GDB Debugger (* LinkServer Debugger (* GUI Flash Tool Other Symbols (* Startup * Source) Common           Installation Commands           Beat and Debug (excend): 3           more inforcement \$londisconnect}           Sload]           Coad mage and Symbon           © Use project binary: lighting-applyxQ2vWk2zwWo.oufidebug/chip-k32w0x-light-example	
□ Java Application	Use file: Workspace File System      Image offset files:         Workspace          File System          Vortes          V	
	As we weapons as.      Insure      Request hardware breakpoint      Resume      Resume	
Filter matched 15 of 15 items	Revert Apply	
3	Debug Close	

- Save the debug configuration by pressing "Apply".
- Set the vector.catch value to false inside the .launch file:
   Right click on the Project -> Utilities -> Open Directory Browser here -> edit \*.launch file:
   <booleanAttribute key="vector.catch" value="false"/>

workspace - MC	CUX presso IDE				
lie Edit Navigat	te Search Project ConfigTools	Run Analysis Fr	reRTOS Window Help		
<b>∷</b> • © ©   ⊗ •	• • • BIO 0 0 X   • I	a = 37 (0, 10, 10)	· · · · · · · · · · · · · · · · · · ·		
Project Explorer	11 🐕 Peripherals+ 💷 Registers	88.01			
<ul> <li>✓ B &gt; matter «D</li> <li>⇒ Ø Projec</li> <li>⇒ Ø Binarie</li> </ul>	efault> Imatter SUF Idevelopment New Go Into	n )		📓 C:\matter\matter LinkServer Default.launch - Notepad++	1
> III, Archiv	Open in New Window Show in Local Terminal	>		File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?	X
> a creder	Copy Paste Delete	Ctrl+C Ctrl+V Delete			
> 🔐 > exar > 🔐 > inter > 🔐 > sorig	Source Move Rename	F2		85 ExmapAttribute key="org.eclipse.debug.core.preferred_launchers">	^
> 🔐 > src 🖾 > 🔐 > third 🖾	Export			"com.nxp.mcuxpresso.core.debug.support.linkserver.launch.LinkServerGdbLaunch"	
BUILD B CODE, E B CONTI B GD, bui	Build Project Clean Project ) Refresh Close Project Close Unrelated Project	FS		/> 87 - 88 <stringattribute key="process_factory_id" value="&lt;/th"><td></td></stringattribute>	
UCEN:	Build Configurations Build Targets Index	>		<pre>"com.nxp.mcuxpresso.core.debug.override.MCXProcessFactory"/&gt; <booleanattribute key="redlink.disable.preconnect.script" value="False"></booleanattribute> double_tribute_tri</pre>	
I REVIEL O	Run As Debug As Profile As	> >		<pre>90 <booleanattribute key="redlink.enable.rlasnashing" value="true"></booleanattribute> 91 <booleanattribute key="redlink.enable.rangestepping" value="true"></booleanattribute> 92 <frigdtribute key="run" value="control"></frigdtribute> 93 </pre>	
- 6	Kestore from Local History Launch Configurations Utilities			93 <booleanattribute key="vector.catch" value="false"></booleanattribute>	
2 Quickstart Pa	SDK Management Tools Validate	- 12	Open directory browser here	94 L 95	
Project W	MCUXpresso Config Tools Run C/C++ Code Analysis	>	<terminated> chip-kl Reconnected to e Connecting to pr</terminated>	eXtensible Marl length : 7,065 lines : 95 Ln : 93 Col : 50 Sel : 5   1 Windows (CR.LF) UTF-8 INS	×
New Binge	Compare With Replace With Configure	>	'Ee(42). Could n Retrying Reconnected to e Server OK but no		
• Build your ( Build Cean	Properties	Alt+Enter	gmene flash con Error: Probe not Error: Probe not Error: Wire not		

- Last step is to start debug the application:



The alternative for this method is to build an SDK example and reuse the debug configuration by selecting the Matter example elf file:

- Import an SDK example:

🔀 SDK Import Wizard					$\times$
The source from the SDK will be copied into the v If you want to use linked files, please unzip the SD	vorkspace. )K_2.x_K32W061DK6' SDF	<u>د</u>			G
🔀 Import projects					
Project name prefix: k32w061dk6	4	Project name suffix			R
Use default location					
Location: C:\Users\nxa29890\Documents\MCUXpre	ssoIDE_11.1.1_3241\work	space\k32w061dk6		Browse	
Project Type		Project Options			
C Project C++ Project C Static Library	C++ Static Library	SDK Debug Console   Semihost   UART   Exa Copy sources Import other files	mple defau	alt	
Examples		<u>ک</u>	u   🖉 🗹 🎙	¥ 🗉 I	8
type to filter					
Name	Descript	ion	Version		^
□ ≡ wireless_uart_bm	The BLE	wireless uart application is a simple demonstration prog.	-		
Image: State	The BLE	wireless uart application is a simple demonstration prog.			
?		< Back Next > Finish		Cancel	

- Change the elf application to debug with the one from the Matter build:

Image: Second	Name: K32w061dk6_wireless_uart_freertos JLink	Debug r Other Symbols) & GUI Flash Tool > Startup & Source Comm
<ul> <li>C/C++ Application</li> <li>C/C++ Attach to Application</li> <li>C/C++ Remote Application</li> <li>C/D+ Remote Application</li> <li>GDB PEMicro Interface Debuggin</li> <li>GDB SEGGER Interface Debuggin</li> <li>GDB SEGGER Interface Debuggin</li> <li>GDB Vardware Debuggin</li> <li>GDB Va</li></ul>	C/C++ Application: /home/@r/matter/examples/lighting-app/nxp/k32w/ Build (if required) before launching Build Configuration: Debug © Enable auto build © Use workspace settings	K32w0/out/debug/chip-k32w0x-light-example VariablesSearch Project

### 4.5 K32W0x1 Matter Application User interface

The example application provides a simple UI that depicts the state of the device and offers basic user control.

This UI is implemented via the general-purpose LEDs and buttons built in to the OM15082 Expansion board attached to the DK6 board:



- Application LED D2 shows the overall state of the device and its connectivity. Four states are depicted:
  - Short Flash On (50ms on/950ms off) The device is in an unprovisioned (unpaired) state and is waiting for a commissioning application to connect.



- Rapid Even Flashing (100ms on/100ms off) The device is in an unprovisioned state and a commissioning application is connected via BLE.
- Short Flash Off (950ms on/50ms off) The device is full provisioned, but does not yet have full network (Thread) or service connectivity.
- Solid On The device is fully provisioned and has full network and service connectivity.
- Application LED D3 shows the state of the simulated light bulb. When the LED is lit, the light bulb is on; when not lit, the light bulb is off.
- Application Button SW2 can be used to reset the device to a default state :
  - A short Press Button SW2 initiates a factory reset.
    - After an initial period of 3 seconds, LED2 D2 and D3 will flash in unison to signal the pending reset. After 6 seconds will cause the device to reset its persistent configuration and initiate a reboot. The reset action can be cancelled by press SW2 button at any point before the 6 second limit.
  - Application Button SW3 can be used to change the state of the simulated light bulb.
- DK6 development board User Interface button
  - A short Press will start advertising for a predefined period;
  - Also, pushing this button starts the NFC emulation by writing the onboarding information in the NTAG.

In case the OM15082 Expansion board is not attached to the DK6 board, like the case in which low power support is enabled, the functionality of LED D2 and LED D3 is taken over by LED DS2, respectively LED DS3, which can be found on the DK6 board. Also, by long pressing the USERINTERFACE button, the factory reset action will be initiated.



### 5 MATTER NETWORK – CHIP TOOL COMISSIONGING AND CONTROL

The prerequisite for this section is to have the i.MX8M Mini + 88W8987 + K32W0x1 setup as OTBR and Matter Controller in order to commission and control Matter devices. The steps for this setup can be found in "*Getting started with i.MX8 Mini in Matter*".

The following instructions will provide information on how to commission and run basic Matter Cluster commands for an application running K32W0x1 platform:

#### Commissioning K32W0x1 over Bluetooth LE:

- Get the operational active dataset from the Thread Network. On the i.mx8m mini run the following command:

\$ ot-ctl dataset active -x



- Activate the BLE advertising on the K32W0x platform by pressing User Interface button;
- Run the following command to commission the device to the Thread Network:

*\$./chip-tool pairing ble-thread {NODE\_ID\_TO\_ASSIGN}{ACTIVE\_DATASET}{SETUP\_PIN\_CODE} {DISCRIMINATOR}* 

Where:

- {NODE\_ID\_TO\_ASSIGN} the matter node id;
- {ACTIVE\_DATASET} Thread operational active dataset. The format is hex:xxxxxx).
- {SETUP\_PIN\_CODE} pin code, default value is 20202021;
- {DISCRIMINATOR} discriminator, default value is 3840;

#### Example:

*\$./chip-tool pairing ble-thread 1* 

#### Sending cluster commands to K32W0x1:

These commands are sent from the imx8m mini shell terminal.

The format of the cluster command is:

\$./chip-tool <cluster\_name> <command\_name> <param1, param2..>



ubu	nt	u@te8:~/connectedhomeip/examples/chip-tool/out/debug\$ ./chip-tool	
[16	44	965778.785693][2756:2756] CHIP:TOO: Missing cluster name	
Usa	ge		
	/ c	hip-tool cluster_name command_name [paraml param2]	
	C.	lusters:	
+		acceggcontro]	-+
		administratorcommissioning	
i		applicationbasic	
i		applicationlauncher	
i		audiooutput	
i		barriercontrol	
		basic	
		binaryinputbasic	
		binding	
		booleanstate	
		bridgedactions	
		bridgeddevicebasic	
		channel	
		colorcontrol	
		contentlauncher	
		descriptor	
		diagnosticlogs	
		discover	
		doorlock	
		electricalmeasurement	
		ethernetnetworkdiagnostics	
		fixedlabel	
		flowmeasurement	
		generalcommissioning	

Example of usage:

Send OnOff cluster -> Toggle command to node ID for cluster endpoint

 chip-tool onoff toggle <node\_id> <cluster\_endpoint>

*\$./chip-tool onoff toggle 11* 

Send OnOff cluster -> On command to node ID for cluster endpoint

 chip-tool onoff on <node\_id> <cluster\_endpoint>

\$./chip-tool onoff on 11

Send OnOff cluster -> Off command to node ID for cluster endpoint
 chip-tool onoff off <node\_id> <cluster\_endpoint>

\$./chip-tool onoff off 11

- Send Attribute reporting configuration for on-off attribute using min/max interval to node ID for cluster endpoint
  - chip-tool onoff report on-off <min\_interval> <max\_interval> <node\_id> <cluster\_endpoint>

\$./chip-tool onoff report on-off 300 301 1 1



Read Channel Attribute from the Thread Diagnostic cluster from node ID

 chip-tool threadnetworkdiagnostics read channel <node\_id> <cluster\_endpoint>

\$./chip-tool threadnetworkdiagnostics read channel 10

- Read fabrics list based on the basic cluster from node ID
  - chip-tool operationalcredentials read fabrics-list <node\_id> <cluster\_endpoint>

\$./chip-tool operationalcredentials read fabrics-list 10



# 6 K32W0X1 - EXPLORING WITH MATTER

### 6.1 Adding Manufacturing data to the application

By default, Matter repository is set to use test certificates found in the credentials folder. They are used in conjunction with default provisioning data and are stored in the embedded application code.

The following workflow describes how to change the certificates and provisioning data to the user's own manufacturing data.

Further details about this feature can be found at <u>https://github.com/NXPmicro/matter/blob/v1.0-branch-nxp/examples/platform/nxp/doc/manufacturing\_flow.md</u>

#### Generate new certificates

This step allows the user to generate a custom Device Attestation Certificate (DAC), Product Attestation Intermediate (PAI) certificate and Product Attestation Authority (PAA) certificate.

This step can be skipped if these certificates are already available from a third-party entity.

Please note that for real production manufacturing data, the "production PAA" certificate authenticity is determined via the Distributed Compliance Ledger (DCL) rather than thorough a PAA certificate generated alongside a DAC and PAI certificate. The PAI certificate may also have a different lifecycle.

As prerequisite for this step, it is required to have the chip-cert compiled in the matter repository. This can be obtained by executing the following commands from the Matter root directory:

\$ source scripts/activate.sh
\$ gn gen out/host
\$ ninja -C out/host

The script to be used for this step is found in the following folder as bash script *generate\_cert.sh*:

<matter\_root>/scripts/tools/nxp/

The output of this script will be a DAC, a PAI and a PAA. User must adjust the PAI\_VID and PAI\_PID values in the script according to their own vendor ID (VID) and product ID (PID).

### \$./scripts/tools/nxp/generate\_cert.sh ./src/tools/chip-cert/out/chip-cert

NXP provides demo certificates for testing with NXP vendor and product IDs that can be found at <matter\_root>/scripts/tools/nxp/demo\_generated\_certs.



This step is to generate new provisioning data and convert all the data to a binary that can be written in the internal flash of the K32W0x1 chip. This example is without any encryption on the data in the binary:

\$ python3 ./scripts/tools/nxp/generate\_nxp\_chip\_factory\_bin.py -i 10000 -s UXKLzwHdN3DZZLBaL2iVGhQi/OoQwIwJRQV4rpEalbA= -p 14014 -d 1000 --dac\_cert /path/to/certs/Chip-DAC-Generated-Cert.der --dac\_key /path/to/certs/Chip-DAC-Generated-Key.der --pai\_cert /path/to/certs/Chip-PAI-Generated-Cert.der --spake2p\_path /out/host/spake2p --out out.bin

Same example as above, but with an already generated verifier passed as input:

\$ python3./scripts/tools/nxp/generate\_nxp\_chip\_factory\_bin.py -i 10000-s UXKLzwHdN3DZZLBaL2iVGhQi/OoQwIwJRQV4rpEalbA= -p 14014 -d 1000--dac\_cert/path/to/certs/Chip-DAC-Generated-Cert.der --dac\_key/path/to/certs/Chip-DAC-Generated-Key.der --pai\_cert/path/to/certs/Chip-PAI-Generated-Cert.der --spake2p\_path ./src/tools/spake2p/out/spake2p --spake2p\_verifier ivD5n3L2t5+zeFt6SjW7BhHRF30gFXWZVvvXgDxgCNcE+BGuTA5AUaVm3qDZBcMMKn1a6CakI4SxyPUnJr0CpJ4pw pr0DvpTlkQKqaRvkOQfAQ1XDyf55DuavM5KVGdDrg== --out out.bin

Here is the interpretation of the parameters:

-i	-> SPAKE2+ iteration
-S	-> SPAKE2+ salt (passed as base64 encoded string)
- <i>p</i>	-> SPAKE2+ passcode
-d	-> discriminator
dac_cert	-> path to the DAC (der format) location
dac_key	-> path to the DAC key (der format) location
pai_cert	-> path to the PAI (der format) location
spake2p_path	-> path to the spake2p tool (compile it from ./src/tools/spake2p)
out	-> name of the binary that will be used for storing all the generated data
aes128_key	-> 128 bits AES key used to encrypt the whole dataset
spake2p_verifier	-> SPAKE2+ verifier (passed as base64 encoded string). If this option is set,
	all SPAKE2+ inputs will be encoded in the final binary. The spake2p tool
	will not be used to generate a new verifier on the fly.



Write the provision data to the internal flash:

Write out.bin to the internal flash of the K32W0x1 chip at location 0x9D200. This can be done using the FTDI USB connection on the DK6 board and the DK6Programmer software. Refer to the Hardware connection in chapter 4.3 for more information on installing the software.

Open a Windows Command Prompt and issue the following command.

C:\nxp\DK6ProductionFlashProgrammer\DK6Programmer.exe -Y-V2-s <COM\_PORT> -P 1000000 -Y-p FLASH@0x9D200="out.bin"

### Generate a new Certificate Declaration (CD)

The script that generates the CD can be found at credentials/test/gen-test-cds.sh. In order to regenerate the CD with the correct data, user needs to update the vids, pid0 and device\_type\_id parameters with the appropriate values. The default Chip-Test-CD-Signing-\* key and certificate can be found at credentials/test/certification-declaration. This CD can act as CSA certificate sand is hard-coded as Trust Anchor in the current chip-tool version. To use this certificate and avoid generating a new one, lines 69-70 must be commented in the gen-test-cds.sh script (the ones that are generating a new CD signing authority).

The command to run the actual CD generation is:

\$./credentials/test/gen-test-cds.sh./src/tools/chip-cert/out/chip-cert

### SET the Vendor and Product IDs

Set the correct VID/PID and CD in the *examples/<example\_app>/nxp/k32w/k32w0/ChipProjectConfig.h* file VID and PID values should correspond to the ones used in the DAC.

The following command can be used to obtain a byte array containing the Certificate Declaration:

\$ hexdump -ve '1/1 "0x%.2x, " path/to/generated/cd/Chip-Test-CD-Generated.der

### **Building the Application**

As stated in the Matter application Building instructions from chapter 4.1, the gn command needs to be adjusted to support the factory data, by adding *chip\_with\_factory\_data=1* to the command.

The command should look like the following:

\$gn gen out/debug --args="k32w0\_sdk\_root=\"\${NXP\_K32W0\_SDK\_ROOT}\" chip\_with\_OM15082=1 chip\_with\_ot\_cli=0 is\_debug=false chip\_crypto=\"tinycrypt\" chip\_with\_se05x=0 chip\_pw\_tokenizer\_logging=true mbedtls\_repo=\"//third\_party/connectedhomeip/third\_party/nxp/libs/mbedtls\" chip\_with\_factory\_data=1"



Build the application using the command:

\$ ninja -C out/debug

#### Running Commissioning using Manufacturer Data

The PAA certificate generated at step 1 in *"Generate new certificates"* needs to be copied to the device that acts as Matter controller. For example, using scp, user can copy with the command:

\$ scp /path/to/generated/certificates/Chip-PAA-Generated-Cert.der root@<ip\_of\_matter\_controller>:/path/to/paa/directory

Open a connection to the Matter controller and run chip-tool with a new PAA:

./chip-tool pairing ble-thread {NODE\_ID\_TO\_ASSIGN} {ACTIVE\_DATASET} {SETUP\_PIN\_CODE} {DISCRIMINATOR} - paa-trust-store-path /path/to/paa/directory

For example:

./chip-tool pairing ble-thread 1 hex:\$hex\_value 14014 1000 --paa-trust-store-path /path/to/paa/directory

Where:

- paa-trust-store-path -> path to the generated PAA (der format). Avoid placing other certificates in the same location as this may confuse chip-tool.
- 14014 is the discriminator
- 1000 is the passcode. This is needed for testing self-generated DACs, but likely not required for "true production" with production PAI issued DACs

Note that the descriminator entered above must match exactly the descriminator value that was selected when the DAC was generated.



# 6.2 Over the air upgrade – Flash Configuration

The support for over-the-air upgrades is enabled by adding chip\_enable\_ota\_requestor to the build command. By default this feature is enabled on the k32w0x1 Matter examples in the args.gni file, otherwise can be enabled like below:

\$ gn gen out/debug --args="k32w0\_sdk\_root=\"\${NXP\_K32W0\_SDK\_ROOT}\" chip\_with\_OM15082=1 chip\_enable\_ota\_requestor=1 chip\_with\_ot\_cli=0 is\_debug=false chip\_crypto=\"tinycrypt\" chip\_with\_se05x=0 chip\_pw\_tokenizer\_logging=true

mbedtls\_repo=\"//third\_party/connectedhomeip/third\_party/nxp/libs/mbedtls\""

### Flash configuration

The internal flash must be prepared for the OTA process:

		Total size: 640K			
SSBL	SSBL Update	Matter application	Factory Data	Meta Data	FlashConfig
8K	8K	614K	2K	1K	8.5K

Total flash size available on K32W0x1 internal memory is 640KB

- SSBL and SSBL update region needed for OTA (mandatory in Matter 1.0) first 16k are reserved for the Bootloader;
- FlashConfig Flash Configuration data that outlines that supported Flash regions, and their corresponding base addresses/sizes. User needs to take care handling this section and not overwrite it!
- Meta Data: e.g.: image signature
- Factory data, including certificates and other security related data
- Matter remains with 614K available for .text (code) and .rodata (constant strings)

The external flash memory is also used for over the air updates and has the following mapping:

#### Maximum size: 16MB

Matter OTA binary	PDM Storage
	Default: 63 x External flash sector size

### 6.3 Reading Tokenizer logs

The debug logging is controlled by the chip\_pw\_tokenizer\_logging build parameter issued at build time. If this is not set or was set to false, then the user can open a serial connection from the host PC with a baudrate of 115200 bps without any changes.

The chip\_pw\_tokenizer\_logging build option can be set to reduce the overall size of the firmware image while enabling debug logging statements by replacing ASCII character strings with encoded (or tokenized) strings.



If chip\_pw\_tokenizer\_logging is set to true, then the detokenizer script must be used. This script decodes the tokenized logs either from a file or from a serial port. The detokenizer script is available in the following path, from the Matter root directory:

examples/platform/nxp/k32w/k32w0/scripts/detokenizer.py

The token database is created automatically after building the binary if the argument *chip\_pw\_tokenizer\_logging*=true was used.

The script can be used in the following ways:

detokenizer.py serial [-h] -i INPUT -d DATABASE [-o OUTPUT]

detokenizer.py file [-h] -i INPUT -d DATABASE -o OUTPUT

Where:

- The first parameter is either serial or file and it selects between decoding from a file or from a serial port.
- *-i INPUT* used to set the path of the file or the serial to decode from.
- *-d DATABASE* represents the path to the token database to be used for decoding. The default path is out/debug/chip-k32w0x-light-example-database.bin after a successful build.
- o OUTPUT represents the path to the output file where the decoded logs will be stored. This
  parameter is required for file usage and optional for serial usage. If not provided when used with serial
  port, it will show the decoded log only at the stdout and not save it to file.

The detokenizer script must be run inside the example's folder after execution of the scripts/activate.sh script. The pw\_tokenizer module used by the script is loaded by the activation script.

An example of running the detokenizer script to see logs of a lighting app:

\$ python3 ../../../examples/platform/nxp/k32w/k32w0/scripts/detokenizer.py serial -i/dev/ttyACM0-d out/debug/chip-k32w0x-light-example-database.bin -o device.txt

## 6.4 ZAP tool. Update Cluster/Endpoint functionality

ZAP - ZCL Advanced Platform is a generic generation engine and user interface for applications and libraries based on <u>Zigbee Cluster Library</u>.

It provides a user interfacefor developers to select specific application configuration (endpoints, clusters, attributes, commands) and perform SDK specific generation of artifacts based on the ZCL specifation and customer provided app configuration.

### Building and running ZAP tool:

It is recomanded to always use the latest version, which at the time of publication of this document is (v16.17.1)

The ZAP tool requires nodejs to be installed, and this cam be done by executing the following commands:



- Download the latest version of <u>nodejs</u>

- Unzip the binary archive to any directory you wanna install Node:

\$sudo mkdir -p /usr/local/lib/nodejs \$sudo tar -xJvf node-v16.15.1-linux-x64.tar.xz -C /usr/local/lib/nodejs

Set the environment variable ~/.profile, add the text below to the end \$sudo nano ~/.profile
 # Nodejs
 VERSION=v16.15.1
 DISTRO=linux-x64
 export PATH=/usr/local/lib/nodejs/node-v16.15.1-linux-x64/bin:\$PATH

- Refresh the profile

\$ . ~/.profile

Running the ZAP-Tool

- Install the zap-tool from the Matter github repositroy:
- Open a terminal and navigate to the Matter root directory

\$cd \*path\_to\_matter\_sdk\*/matter

\$source ./scripts/activate.sh

\$./scripts/tools/zap/run\_zaptool.sh

- The first time this script is executed will take longer, as zap-tool is being installed
- If the installation is succesfull you will see the below feedback on the terminal and ZAP-Tool will open:

Edit View Wiedow Hele		New Configuration				-	•
zcl.			GENERATE	REGENERATE	PREVIEW	٥	0
luster Configurator: N	Aatter S	DK ZCL data	S DISPLAYBUTTON ZCL	GLOBAL OPTIONS.	. ≔ ZCL	extensio	NS
+ ADD NEW ENDPOINT							
		ZCL Advanced Platform					
		() Generation and back and tests (passing () Cypress UI tests (passing () SonarCloud (passing () Build &	Release passing				
		What is ZAP?					
		ZAP is a generic generation engine and user interface for applications and libraries based on Zigbee	Cluster Library, the specification developed by t	ne <u>Zigbee Alliance</u> .			
		ZAP allows you to perform the following:					
		<ul> <li>perform SDK specific customized generation of all global artifacts (constants, types, IDa, etc.)</li> <li>provide UI for the end-user to select specific application configuration (clusters, attributes, cor</li> <li>perform SUK specific customized generation of all user selected configuration artifacts (applic provided application configuration.</li> </ul>	based on the ZCL specification nmands, etc.) cation configuration, endpoint configuration, etc	based on ZCL spe	cification and c	ustomer-	
		ZAP is a generic templating engine. Examples are provided for how to generate artifacts for the C lan such as C++, java, node js, python or any other.	nguage environment, but one could easily add ne	w templates for oth	er language ei	nvironment	s,
		Quick instructions					

 Next, open the ZAP file for the corresponding project. For Lighting-app -> Go to .../matter/examples/lighting-app/lighting-common and select the lighting-app.zap file:



		③ Recent			matter	examples	lighting-app	lighting-common	•					
it View Wir	ndow I	G Home	Name							^	Size	Туре	Modified	
		Desktop	src										oL	
CL			lighting-app.za	p							241.4 kB	Text	oL	eview 🌣
	-		c lighting-app.ma	atter							62.3 kB	Text	oL	
ster Confi	igurat	Downloads	BUILD.gn								1.3 kB	Text	oL	E ZCL EXTENS
- ADD NEW	ENDPOI	Music												_
		Pictures												
point - 0	DELET	⊟ Videos												
ce	Matter	🗅 геро												
vork	(0x001 0	+ Other Locations												
ile ID	0x010													Config
ion	1													
oled Clusters	42													- <b>\$</b>
oled Attributes	246													
oled Reporting	250													<b></b>
point - 1	DELETR													- 0
ice	Matter													
Nork	Light (													
ile ID	0x010													- ×

- A new ZAP-tool window will open with the endpoint/cluster configuration for the selected application:

			moneye	amprest righting appling in	cing-commony agric	ing-opp.rop				
e Edit View Win	ndow Help									
ZCL							GENERATE	REGENERATE PR	EVIEW	¢ 0
Cluster Confi	igurator: Matter SDK	( ZC	L data				AYBUTTON ZCL GLO	BAL OPTIONS	I ZCL	EXTENSIONS
+ ADD NEW I	ENDPOINT ┥ 🕇	Enc	dpoint 1 Clusters							
Endpoint - 0	DELETE 🧪 EDIT 🛧		-							
Device	Matter Root Node	Show	w No Filter 👻 CLOSE ALL				Q Search Clust	ers		
Network	0	^	General							
Profile ID	0x0103		Cluster	Required Cluster	Cluster ID	Manufacture	er Code Enable			Configure
Version	1								_	
Enabled Clusters	42		Identify	Server	0x0003		Server		•	•
Enabled Attributes	246		Groups	Server	0x0004		Server			•
Enabled Reporting	250						ociver			×
Endpoint - 1	DELETE 🧪 EDIT 🛧	▲	Scenes	Server	0x0005	-	Not En	abled	•	\$
Device	Matter Dimmable Light (0x0101)		On/Off	Server	0x0006		Server		•	٠
Network Profile ID	0 0x0103		On/off Switch Configuration		0x0007	-	Not En	abled	•	\$
Version	1		Level Control	Server	0x0008		Server		•	٠
Enabled Attributes	100		Binary Input (Basic)		0x000F	-	Not En	abled	•	\$
Enabled Reporting	137									

- After performing all the changes, generate the new configuration -> by pressing the *Generate* button and save the new configuration in the .../matter/zzz\_generated/lighting-app/zap-generated folder:



	Cancel			Q	Generate	
t View Window I	③ Recent	Generated matter zzz_generated lighting-app zap-generated →			E2	
	습 Home	Name	⊾ Size	Туре	Modified	-
	Desktop	PluginApplicationCallbacks.h     IMClusterCommandHandler.cpp	4.4 kB 51.4 kB	Text Text	oL oL	EVOEN .
t View Window I	🖻 Documents	gen_tokens.h	1.9 kB	Text	Jo	
CL.		b gen_config.h i endpoint_config.h	9.5 kB 94.9 kB	Text Text	oL	
	Music	CHIPClusters.h	1.4 kB	Text	Jo	
ator Configurat	JJ Music	CHIPClientCallbacks.h	1.1 kB	Text	Jo	
ster configurat	Pictures	af-gen-event.h	1.4 kB	Text	Jo	= ZULENI
	⊟ Videos	access.h	22.1 kB	Text	Jo	
- ADD NEW ENDPOI	🗀 геро					
point - 0 📋 DELI	+ Other Locations					с
point - 1 🥛 DELL						-
						•
						•
						-
						•

- Rebuild the application to apply the new configuration.