AN1384: Over-the-Air Bootload Server and Client Setup for Zigbee SDK 7.0 and Higher

This application note describes the process users should follow to perform a Zigbee® OTA (over-the-air) bootloading session between a ZCL OTA Upgrade cluster client device and server device. The hardware used is an EFR32MG12-based development kit. (See the WSTK6001 User Manual for details about how to configure the WSTK development board for use with the EFR32MG12.) You can also refer to procedures in this document when setting up or testing OTA bootload downloads in your own development environments with your own hardware.

If you are developing with the EmberZNet SDK 6.10.x and lower, see *AN728: Over-the-Air Bootload Server and Client Setup*.

Before you begin implementing the procedures in this document, you should be familiar with the basics of building and loading application images on Silicon Labs platforms. See *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher* for details.

KEY POINTS

- Configure and build two Zigbee OTA client images.
- Configure and build the OTA server.
- Load the original client image.
- Load the NCP server software.
- Load the updated OTA client image on the server.

1 Introduction

This application note describes the steps necessary to demonstrate a Zigbee OTA bootloading session between a ZCL OTA Upgrade cluster client device and server device. The procedures use EFR32MG12-based development boards. (See the WSTK6001 User Manual for details about how to configure the WSTK development board for use with the EFR32MG12.)

Refer to *UG103.6: Bootloading Fundamentals* for more details about the bootloader. For details about the ZCL OTA Upgrade cluster, consult the latest published revision of the Zigbee cluster library specification. Also note that, in the Zigbee Cluster Configurator, the ZCL OTA Upgrade cluster is referred to as OTA Bootloading. For consistency, that terminology is used in this document as well. If you are searching Zigbee documents, be sure to search for ‘OTA Upgrade.

Refer to this application note when setting up and or testing the Zigbee OTA bootload cluster on your own. The Advanced section describes how this example can be expanded to use your own hardware configuration, and how to change the software configuration to support your own manufacturer-specific information.

For further reference, see:

- Zigbee Document #07-5123, *Zigbee Cluster Library Specification* - “Over-The-Air Upgrade” chapter; available from https://csa-iot.org/.
- *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher*
- *UG491: Zigbee Application Framework Developer's Guide for SDK 7.0 and Higher*
- *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*
- *AN1084: Using the Gecko Bootloader with EmberZNet*
- *AN1325: Zigbee Cluster Configurator User's Guide*
- *AN1389: Running Zigbee Host Applications in a Docker Container*

2 Hardware and Software Requirements

Many of the steps in this document are performed using the project configuration tools provided by Simplicity Studio 5 (SSv5). These tools include a Project Configurator, a Component Editor, and a Zigbee Cluster Configurator. For more information on configuring an application, refer to *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher*, provided with your release.

Several of the procedures use IAR-EWARM as the compiler. The IAR-EWARM version must be compatible with the EmberZNet SDK version. See the release notes for the SDK to determine the compatible compiler version number.

These procedures use two EFR32MG12-based WSTKs (Wireless Starter Kits), one for the client device running in System-on-Chip (SoC) mode, and one as the Network Co-Processor (NCP) component of the OTA Server. A PC Host running Linux or Windows Docker container is also required as part of the OTA Server setup.

2.1 Client Hardware

The client device used in these procedures is an EFR32MG12-based WSTK running in SoC mode. You have two choices for the download space. The first is to use an external storage device, such as a serial dataflash or serial EEPROM, connected to the device. The second option, only available with devices with more than 512 kB flash space, is to use a portion of the main flash as the download space. Bootloaders using the second option are often referred to as either internal or local storage bootloaders. EFR32MG12 parts can use local storage; EFR32MG1 parts cannot. For a full list of supported external memory parts for EFR32MGx parts, see *UG489: Silicon Labs Gecko Bootloader User Guide*.

For devices using the EFR32MG1, only the serial dataflash option is available; no local storage application bootloader is presently offered. However, parts whose numbers begin with EFR32MG1x6 or EFR32MG1x7 contain an integrated serial flash that can be used just like off-chip serial dataflash but without any additional components.

EFR32MG12-based or MG13-based devices with at least 512 kB of internal flash also support internal storage for use with a local storage application bootloader design, as well as external storage for use with an SPI flash-based application bootloader design. In the procedures described in this document, internal storage is used on an EFR32MG12 with 1024 kB of flash. Contact Silicon Labs technical support for assistance with other memory layouts for other devices.

The Zigbee OTA client cluster can be used with an EmberZNet PRO-based device running in either SoC or EZSP NCP mode. This document does not show how to configure an EZSP host application for an NCP-based client, but the server hardware and setup described in section 2.2 Server Hardware is the same when the client is using EZSP. This application note shows one example of how the OTA client cluster can be used with a specific hardware configuration.

The client software is based upon the ZCL Application Framework contained within the latest version of the EmberZNet PRO stack.

2.2 Server Hardware

The NCP device used in these procedures is an EFR32MG-based WSTK with a UART-connected host application communicating via EmberZNet Serial Protocol (EZSP). In this application note, either a Linux system or Windows PC with Docker container is required to run the host software that connects to the NCP. The storage for the OTA software images is the host's local file system. To avoid linking issues, do not use a virtual machine to run the host.

It is possible for an OTA server to run on an EFR32MG-based SoC, or to use an NCP with a different (non-POSIX-based) host system. A different configuration than the one presented here requires an alternative mechanism for pushing images into the OTA server so they can then be served up to clients through the Zigbee OTA bootload cluster protocol. For example, software images could be pushed through a utility backhaul, an Ethernet connection to a local network, or some other proprietary mechanism.

A Linux system connected to a development board acting as a UART NCP is only one possible option to serve up OTA files. It was the friendliest option for an OTA server because there are many mechanisms to push OTA files into the server's file system.

The following figure shows a diagram of the hardware configuration used for the Zigbee OTA application bootload procedures. Note that only one client is required.

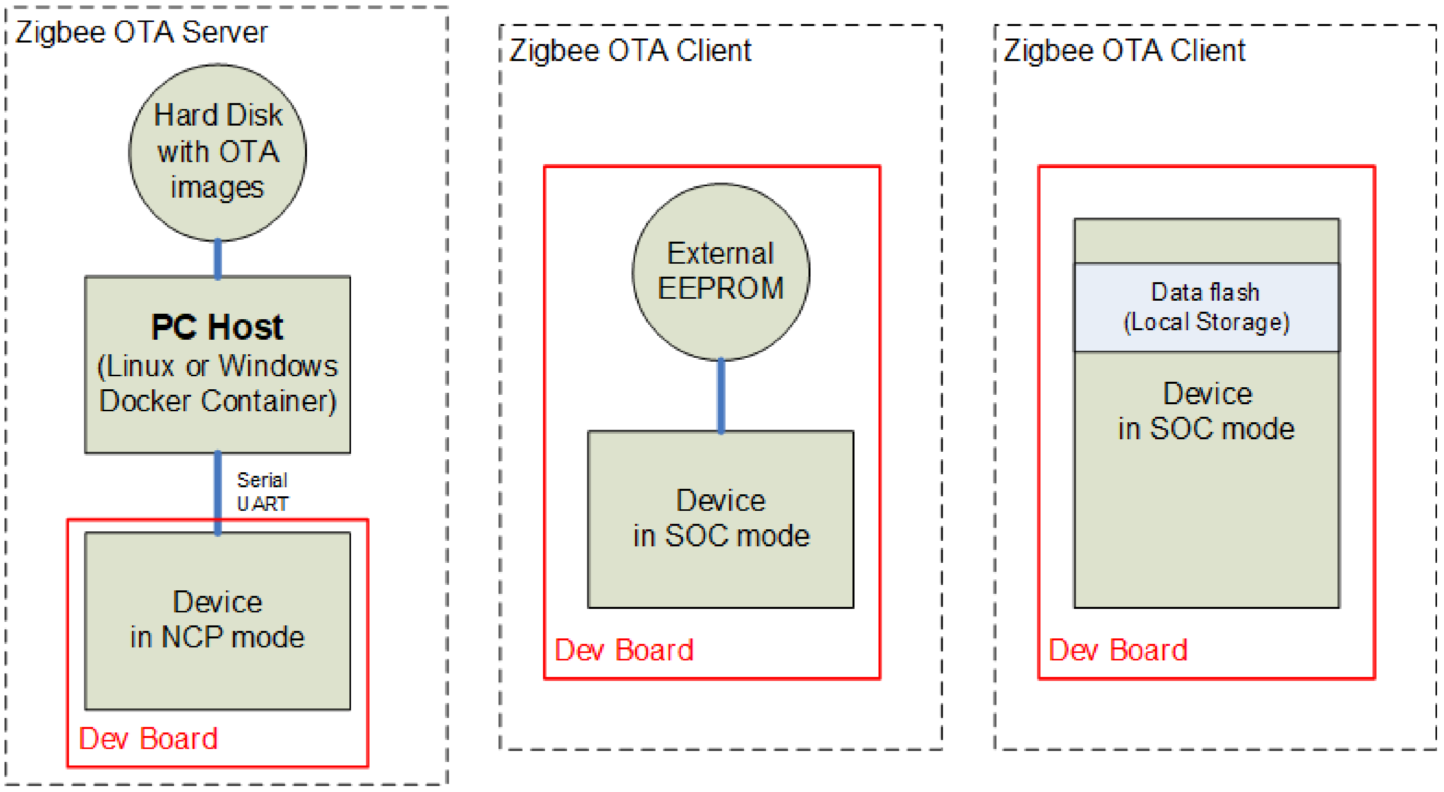


Figure 2.1. Zigbee OTA Application Bootload Hardware Diagram

3 OTA File Storage

3.1 About the Zigbee OTA File Format

The OTA file format is defined in the *Zigbee Cluster Library Specification*, in the “OTA File Format” section of the “Over-The- Air Upgrade” chapter. Images are composed of a Zigbee OTA header, followed by one or more blobs of proprietary bootloader data, and then an optional set of cryptographic signature data appended to the end. The OTA server only needs to read the OTA header to serve up the file, and thus can serve up files for different manufacturers and or different products. The following figure shows an example of the layout of the file format.

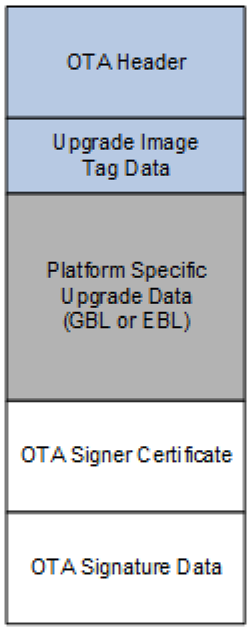


Figure 3.1. File Format Sent Over-the-Air

Silicon Labs has created a tool for generating OTA images called **Image-builder**. The non-ECC version of **Image-builder** is included with the EmberZNet PRO release in the **tool\image-builder** sub-directory. For more information on Image-builder, see *AN716: Instructions for Using Image Builder*.

OTA signing refers to signing the OTA image file and is independent of any signature or encryption that may be present on the update image file inside the OTA wrapper file. Currently, only the Smart Energy Profile has support for signatures on images. This application note assumes that OTA images do not need to be signed.

3.2 About OTA File Storage

The new image is saved in external flash or internal flash, depending on which bootloader is used.

- If using an SPI Storage Bootloader, the new image will be saved into SPI flash.
- If using an Internal Storage Bootloader, the new image will be saved into internal flash.

The offset of the new image is set in the **OTA Simple Storage EEPROM Driver** software component. There are two typical settings.

1. Using slot-manager: The slot offset is defined in the bootloader. In the application, set **Gecko Bootloader Storage Support** to any selection except **Do not use slot**. In this case, the **OTA Storage Start Offset** and **OTA Storage End Offset** will not be used.
2. Using absolute offset: In the application, you must set **Gecko Bootloader Storage Support** to **Do not use slots**. In this case, the **Storage Slot To Save Images To** will not be used. The **OTA Storage Start Offset** and **OTA Storage End Offset** must be set correctly. The **OTA Storage Start Offset** must equal the start offset of a slot defined in the bootloader. The **OTA Storage End Offset** must equal the start offset plus the slot size of that slot.

4 General Procedure

The following outlines the steps to configure the OTA bootload cluster client and server. Details are presented in subsequent sections.

1. Configure and build the original OTA client image.
2. Configure and build the updated OTA client image.
3. Create the Zigbee OTA server files.
4. Build the PC host.
5. Build Internal Storage Bootloader.
6. Load the original client image.
7. Load the EZSP NCP server software.
8. Load the updated OTA client image on the server.
9. Run the Zigbee OTA client / server.

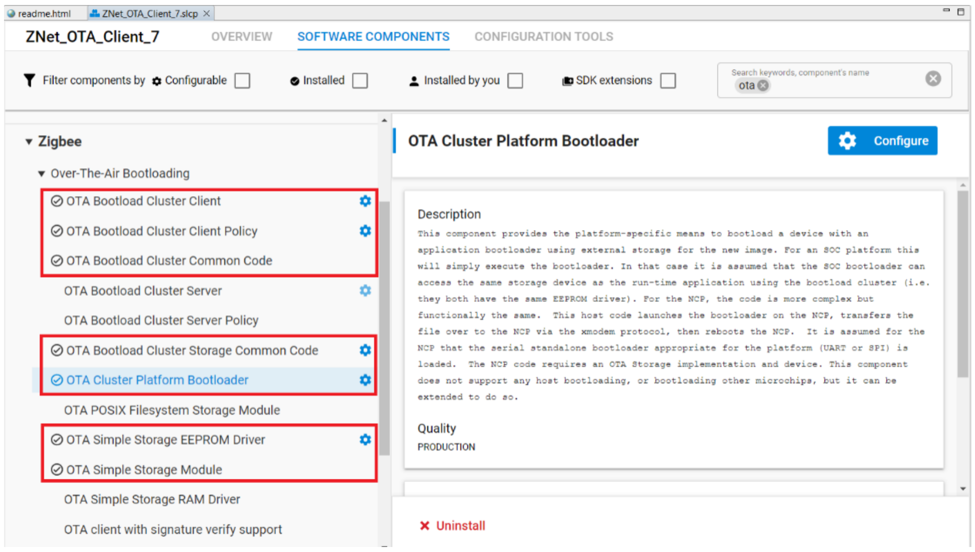
4.1 Configure and Build the Original Zigbee OTA Client Image

This procedure configures a Zigbee LO On/Off Light (Zigbee – SoC Light) to run on the BRD4161A board as an example, but it can be adapted for any sample application. Other Zigbee device types can be configured similarly. The example setup includes support for the Zigbee OTA bootload cluster client.

The procedures in this application note build two client images: the original version and the updated version. This client will be the original version. The image generated will run before the Zigbee OTA process upgrades the client device.

Perform the following steps to configure the OTA client:

1. In Simplicity Studio, create and name a new project. This can be blank or based on one of the provided examples. For the purposes of this exercise, you may want to name the file **ZNet_OTA_Client_7** (for the 7.x.x stack).
2. Open the Project Configurator and click on the **SOFTWARE COMPONENTS** tab.
3. Search for OTA-related software components and install the following highlighted components.


readme.html
ZNet_OTA_Client_7.slcp
ZNet_OTA_Client_7
OVERVIEW
SOFTWARE COMPONENTS
CONFIGURATION TOOLS
Filter components by
Configurable
Installed
Installed by you
SDK extensions
Search keywords, component's name
ota
Zigbee
Over-The-Air Bootloading
OTA Bootload Cluster Client
OTA Bootload Cluster Client Policy
OTA Bootload Cluster Common Code
OTA Bootload Cluster Server
OTA Bootload Cluster Server Policy
OTA Bootload Cluster Storage Common Code
OTA Cluster Platform Bootloader
OTA POSIX Filesystem Storage Module
OTA Simple Storage EEPROM Driver
OTA Simple Storage Module
OTA Simple Storage RAM Driver
OTA client with signature verify support
Configure
Uninstall

OTA Cluster Platform Bootloader

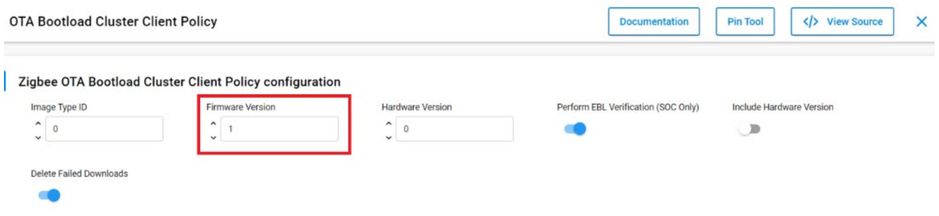
Description

This component provides the platform-specific means to bootload a device with an application bootloader using external storage for the new image. For an SOC platform this will simply execute the bootloader. In that case it is assumed that the SOC bootloader can access the same storage device as the run-time application using the bootload cluster (i.e. they both have the same EEPROM driver). For the NCP, the code is more complex but functionally the same. This host code launches the bootloader on the NCP, transfers the file over to the NCP via the xmodem protocol, then reboots the NCP. It is assumed for the NCP that the serial standalone bootloader appropriate for the platform (UART or SPI) is loaded. The NCP code requires an OTA Storage implementation and device. This component does not support any host bootloading, or bootloading other microchips, but it can be extended to do so.

Quality

PRODUCTION

4. Configure the **OTA Bootload Cluster Client Policy** component by setting the **Firmware Version** to a value of **1** as the original version number. This is critical, as later steps assume you have the version set to a **1**.

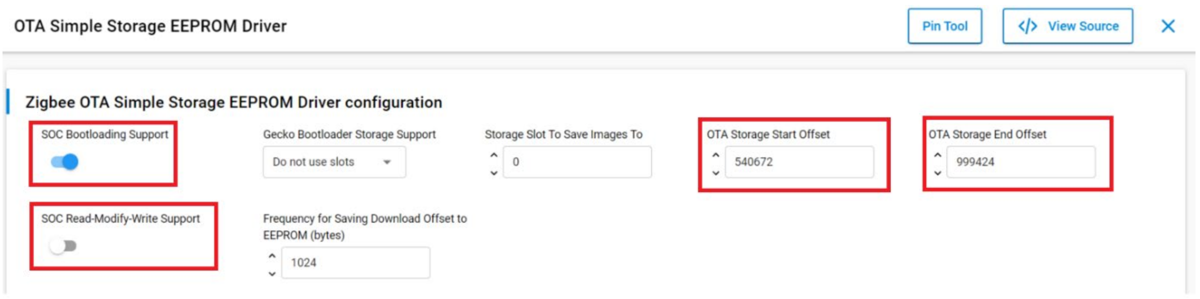


OTA Bootload Cluster Client Policy

Zigbee OTA Bootload Cluster Client Policy configuration

Image Type ID	0
Firmware Version	1
Hardware Version	0
Perform EBL Verification (SOC Only)	On
Include Hardware Version	Off
Delete Failed Downloads	On

5. Configure the **OTA Simple Storage EEPROM Driver** component by enabling the **SOC Bootloading Support** and disabling **SOC-Read-Modify-Write Support**. The **OTA Storage Start Offset** and **OTA Storage End Offset** values in this component may need to be changed for other storage layouts, including layouts involving Storage Slots. Contact Silicon Labs technical support for guidance on choosing other values based on your use case. For this example, the OTA client’s **OTA Storage Start Offset** (540672) and the **End Offset** (999424) are configured to match those specified for the Internal Storage Bootloader to be built in the later section.

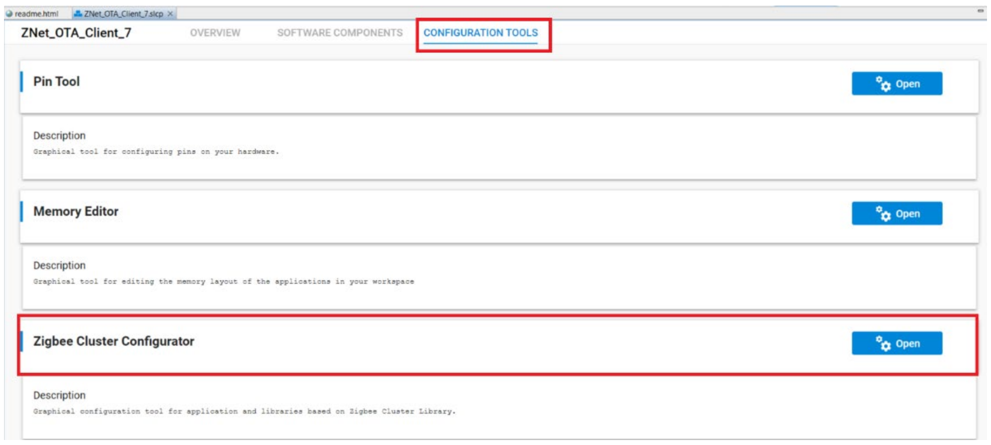


OTA Simple Storage EEPROM Driver

Zigbee OTA Simple Storage EEPROM Driver configuration

SOC Bootloading Support	On
Gecko Bootloader Storage Support	Do not use slots
Storage Slot To Save Images To	0
OTA Storage Start Offset	540672
OTA Storage End Offset	999424
SOC Read-Modify-Write Support	Off
Frequency for Saving Download Offset to EEPROM (bytes)	1024

6. Now Open the Project Configurator and select the **CONFIGURATION TOOLS** tab. Open the **Zigbee Cluster Configurator**.



Pin Tool

Description
Graphical tool for configuring pins on your hardware.

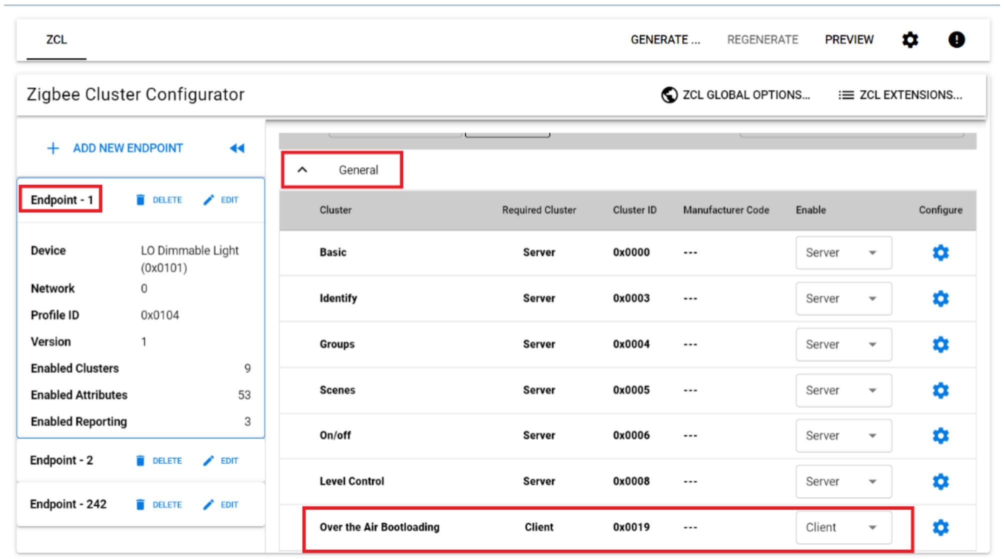
Memory Editor

Description
Graphical tool for editing the memory layout of the applications in your workspace

Zigbee Cluster Configurator

Description
Graphical configuration tool for application and libraries based on Zigbee Cluster Library.

7. Highlight the **Endpoint-1** from the Cluster List on the left pane. Expand the **General** group and enable the **Over-the-Air Bootloading Cluster** client as shown below. Save the change with shortcut key CTRL-S or File->Save.



Endpoint - 1

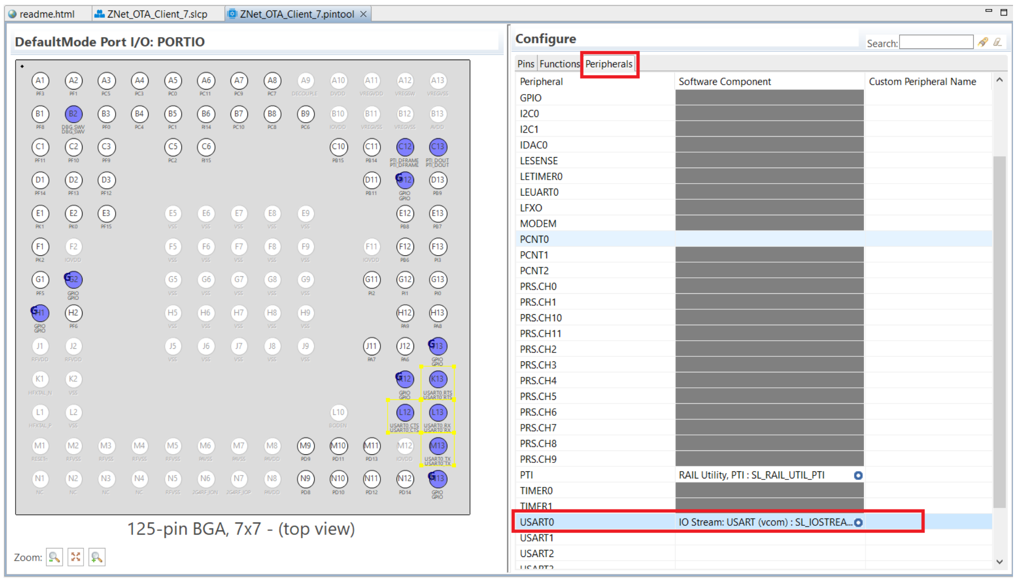
Device	LO Dimmable Light (0x0101)
Network	0
Profile ID	0x0104
Version	1
Enabled Clusters	9
Enabled Attributes	53
Enabled Reporting	3

Endpoint - 2

Endpoint - 242

Cluster	Required Cluster	Cluster ID	Manufacturer Code	Enable	Configure
Basic	Server	0x0000	---	Server	
Identify	Server	0x0003	---	Server	
Groups	Server	0x0004	---	Server	
Scenes	Server	0x0005	---	Server	
On/off	Server	0x0006	---	Server	
Level Control	Server	0x0008	---	Server	
Over the Air Bootloading	Client	0x0019	---	Client	

8. Open the Project Configurator and select the **CONFIGURATION TOOLS** tab. Open the **Pin Tool** and select the **Peripherals** tab on the **Configure** pane on the right. Verify that the **USART0** has the **IO Stream: USART (vcom)** selected as its software component. Save any changes made to Pin Tool with shortcut key CTRL-S or File->Save.


readme.html
ZNet_OTA_Client_7.slcp
ZNet_OTA_Client_7.pintool
DefaultMode Port I/O: PORTIO
125-pin BGA, 7x7 - (top view)
Zoom:
Configure
Search:
Pins
Functions
Peripherals

Peripheral	Software Component	Custom Peripheral Name
GPIO		
I2C0		
I2C1		
IDAC0		
LESENSE		
LETIMER0		
LEUART0		
LFXO		
MODEM		
PCNT0		
PCNT1		
PCNT2		
PRS.CH0		
PRS.CH1		
PRS.CH10		
PRS.CH11		
PRS.CH2		
PRS.CH3		
PRS.CH4		
PRS.CH5		
PRS.CH6		
PRS.CH7		
PRS.CH8		
PRS.CH9		
PTI	RAIL Utility, PTI : SL_RAIL_UTIL_PTI	
TIMER0		
TIMER1		
USART0	IO Stream: USART (vcom) : SL_IOSTREA...	
USART1		
USART2		

9. Build the application according to your preferred method. If a .gbl (Gecko Bootloader) file is not generated as part of the build, you may run the following commander command to create it manually from the S-record file `ZNet_OTA_Client_7.s37` as shown below. More information about the commander can be found in *UG162: Simplicity Commander Reference Guide*.

```
commander.exe gbl create .\ZNet_OTA_Client_7.gbl --app .\ZNet_OTA_Client_7.s37
```

4.2 Configure and Build the Updated Zigbee OTA Client Image

This section builds the updated version image. The images are identical except for the version number embedded in them.

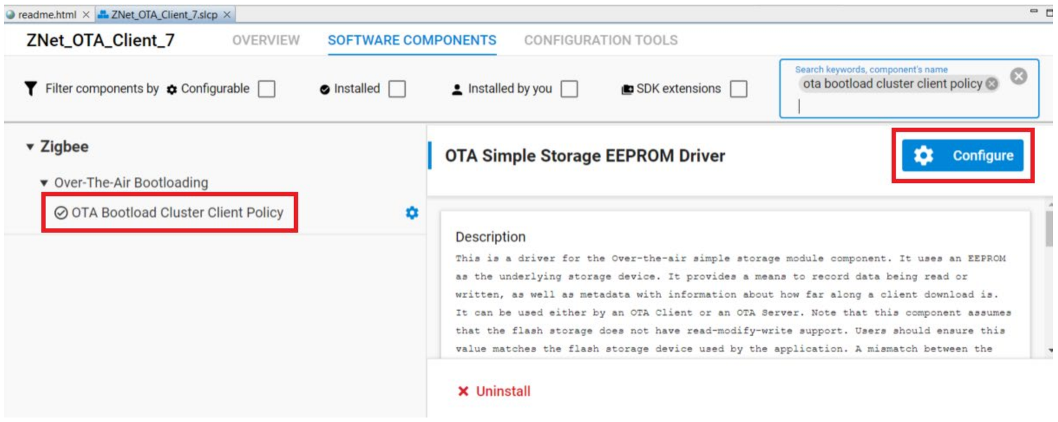
4.2.1 Copy the Original OTA Client Image

Copy the original `ZNet_OTA_Client_7.gbl file` with its **Firmware Version** set to **1** to another folder to avoid it being overwritten when the "updated" version is built next to enable an OTA update. Rename this file appropriately, for example: `ZNet_OTA_Client_7_v1.gbl`.

4.2.2 Modify, Generate, and Build the Updated OTA Client Configuration

To generate a new OTA Client configuration, perform the following steps.

1. Open the previously saved OTA Client project created in section 4.1 Configure and Build the Original Zigbee OTA Client Image.
2. Open the Project Configurator and click on the **SOFTWARE COMPONENT** tab.
3. Search for the **OTA Bootload Cluster Client Policy** component and click **Configure** to configure it.



This is a driver for the Over-the-air simple storage module component. It uses an EEPROM as the underlying storage device. It provides a means to record data being read or written, as well as metadata with information about how far along a client download is. It can be used either by an OTA Client or an OTA Server. Note that this component assumes that the flash storage does not have read-modify-write support. Users should ensure this value matches the flash storage device used by the application. A mismatch between the

4. Change the value in **Firmware Version** to **22**. This value is an arbitrary choice, but it must be greater in value than the previous version.

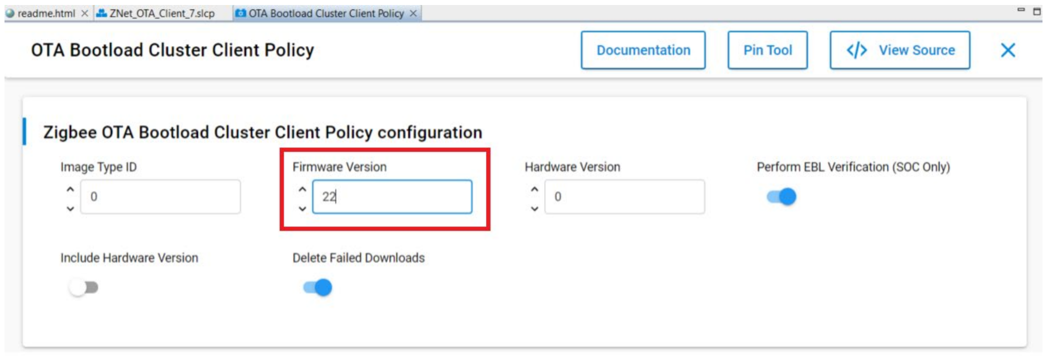

readme.html
ZNet_OTA_Client_7.slcp
OTA Bootload Cluster Client Policy
OTA Bootload Cluster Client Policy
Documentation
Pin Tool
View Source
Zigbee OTA Bootload Cluster Client Policy configuration

Image Type ID: 0
Firmware Version: 22
Hardware Version: 0
Perform EBL Verification (SOC Only): on
Include Hardware Version: off
Delete Failed Downloads: on

5. Build the project. Locate the `.ota` file created in the compiler output directory. However, if only the `.s37` file is created from the build, then follow the below instruction to create the file `ZNet_OTA_Client_7.ota` needed for later upgrade. More information about the image-builder utility may be found in *AN716: Instructions for Using Image-Builder*.

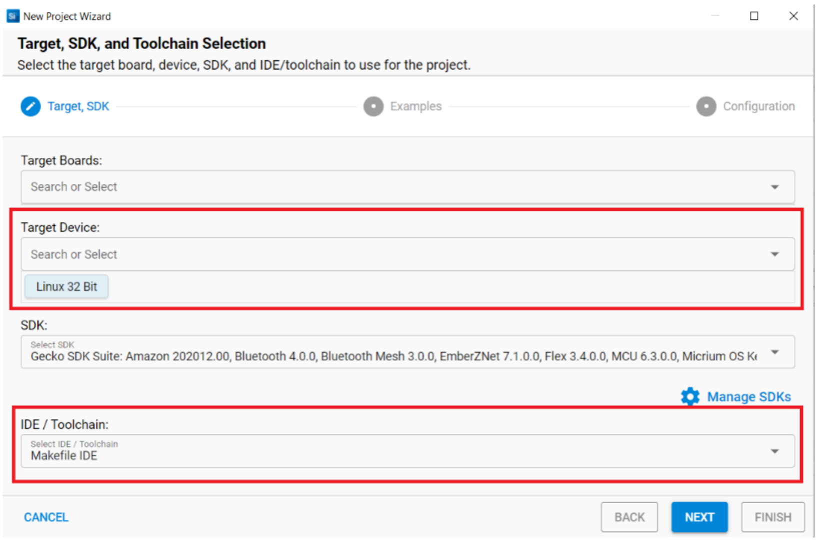
```
commander.exe gbl create .\ZNet_OTA_Client_7.gbl --app .\ZNet_OTA_Client_7.s37

image-builder-windows.exe --create ZNet_OTA_Client_7.ota --version 22 --manuf-id 0x1002 --image-type 0 --tag-id
0x0000 --tag-file .\ZNet_OTA_Client_7.gbl --string "ZNet_OTA_Client_7"
```

6. Copy `ZNet_OTA_Client_7.ota` to the directory where you saved your `ZNet_OTA_Client_7_v1.gbl` and rename it to track the version number, for example `ZNet_OTA_Client_7_v22.ota`.

4.3 Create the Zigbee OTA Server Files

This procedure creates the source for a Zigbee 3.0 gateway host device. The device includes support for the Zigbee OTA bootload cluster server. The host server software can support both signed and unsigned OTA images since it will not do any verification on these files. To configure the OTA server:

1. In Simplicity Studio, start the New Project Wizard. Select **Linux 32 Bit** as the target device and **Makefile IDE** as the Toolchain. Click **Next**.



New Project Wizard

Target, SDK, and Toolchain Selection

Select the target board, device, SDK, and IDE/toolchain to use for the project.

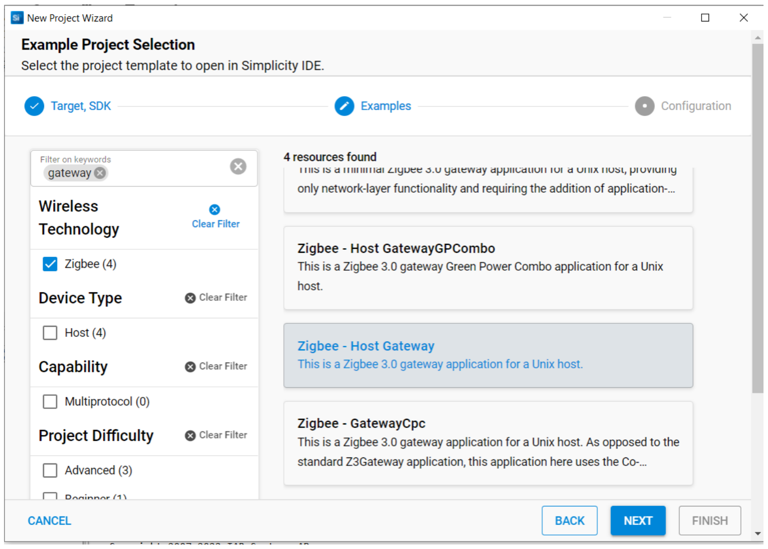
Target Boards: Search or Select

Target Device: Search or Select
Linux 32 Bit

SDK:
Select SDK
Gecko SDK Suite: Amazon 202012.00, Bluetooth 4.0.0, Bluetooth Mesh 3.0.0, EmberZNet 7.1.0.0, Flex 3.4.0.0, MCU 6.3.0.0, Micrium OS K

IDE / Toolchain:
Select IDE / Toolchain
Makefile IDE

2. Select the Zigbee Host Gateway as the example project as shown below and click **Next**.



New Project Wizard

Example Project Selection

Select the project template to open in Simplicity IDE.

Filter on keywords: gateway

Wireless Technology — Clear Filter
☑ Zigbee (4)

Device Type — Clear Filter
☐ Host (4)

Capability — Clear Filter
☐ Multiprotocol (0)

Project Difficulty — Clear Filter
☐ Advanced (3)
☐ Beginner (1)

4 resources found

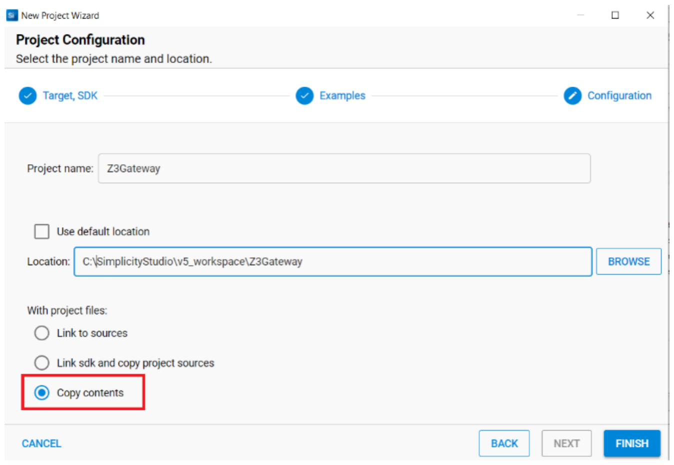
This is a minimal Zigbee 3.0 gateway application for a Unix host, providing only network-layer functionality and requiring the addition of application-...

Zigbee - Host GatewayGPCombo
This is a Zigbee 3.0 gateway Green Power Combo application for a Unix host.

Zigbee - Host Gateway
This is a Zigbee 3.0 gateway application for a Unix host.

Zigbee - GatewayCpc
This is a Zigbee 3.0 gateway application for a Unix host. As opposed to the standard Z3Gateway application, this application here uses the Co-...

3. Select **Copy contents** and click **Finish** to create the project as shown below.



Project name: Z3Gateway

Use default location [ ]

Location: C:\SimplicityStudio\v5_workspace\Z3Gateway

With project files:
Link to sources
Link sdk and copy project sources
[✓ Copy contents]

4. The Z3Gateway host project files are generated in the Location specified in the previous step.

4.4 Build the PC Host

Building the OTA server can be done on a Linux system or in the Windows Docker container. This application note assumes that you are NOT using a cross-compiler and that the target system is the same system as the one where the development tools exist. It is possible to cross-compile for a different Linux system (for example, on an x86 PC targeting an embedded ARM Linux system). See section 5.4 Using a Cross-Compiler for the OTA Server for details on how to cross-compile for other systems.

Building the OTA Server for Linux requires several development tools. We assume you have access to these tools. They are as follows:

- Make
- GCC
- sed
- awk
- xargs
- The standard C Library and its development headers
- The Readline Library and its development headers
- The Ncurses Library and its development headers

The steps to build the OTA server under Linux follow. It is assumed that these steps are run after configuring the server in Simplicity Studio.

1. Copy the EmberZNet PRO stack files from their installed location on Windows to a directory on the Linux PC (e.g. User Home directory). This should include the files that were generated in a previous step.
2. Launch a Bash Shell.
3. Change to the EmberZNet project directory on your Linux PC or Raspberry Pi. For example, `cd /home/<UserID>/Z3Gateway`
4. Run **Make** on the generated Makefile from that directory, for example `make -f Z3Gateway.Makefile` or simply `make` if the generated Makefile is in the current working directory and is specifically named **Makefile**.
5. The compilation should complete successfully with the generated `Z3Gateway` executable in the subdirectory `./build/debug/`

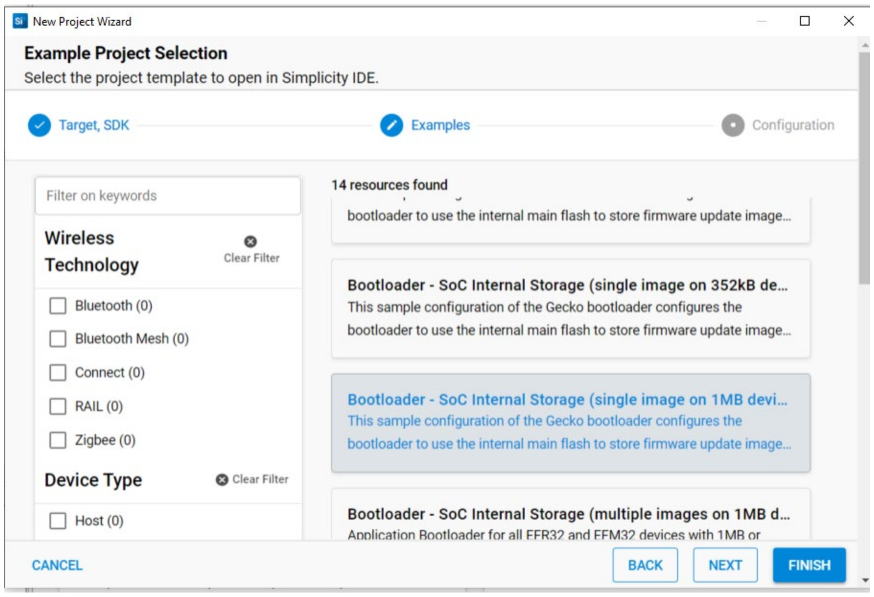
To build the Z3Gateway OTA Server using the Windows Docker container, see *AN1389: Running Zigbee Host Applications in a Docker Container*.

4.5 Build Internal Storage Bootloader

To run OTA upgrade successfully, the bootloader must be selected and configured correctly. Review *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* to learn more about how the bootloader works and how to configure it for OTA upgrade.

The following steps configure and build an Internal Storage Bootloader that stores firmware update image on an EFR32MG12 of a BRD4161A board. The storage slot is configured to start at address 0x84000 with the size of 448 kB. This bootloader can be used by both the OTA client and server in the subsequent section.

1. Run the **New Project Wizard** from Simplicity Studio (SSv5) to select and create the new internal storage bootloader project.



14 resources found

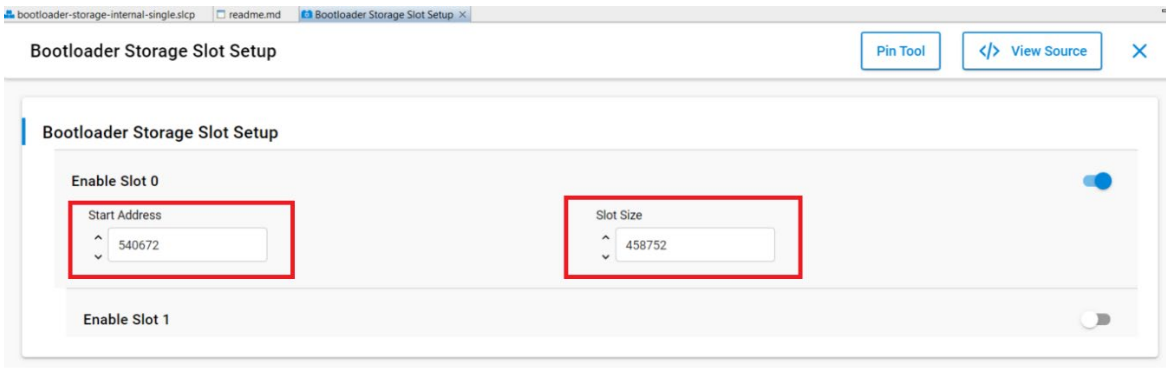
bootloader to use the internal main flash to store firmware update image...

Bootloader - SoC Internal Storage (single image on 352kB de...
This sample configuration of the Gecko bootloader configures the bootloader to use the internal main flash to store firmware update image...

Bootloader - SoC Internal Storage (single image on 1MB devi...
This sample configuration of the Gecko bootloader configures the bootloader to use the internal main flash to store firmware update image...

Bootloader - SoC Internal Storage (multiple images on 1MB d...
Application Bootloader for all EFR32 and EFM32 devices with 1MB or

2. Once the project is created, open and configure the Bootloader Storage Slot Setup software component. Make sure the storage start address is at 540672 (0x84000) with a size of 458752 (448kB). Make sure these values match with the OTA client's **OTA Storage Start Offset** (540672) and the **End Offset** (999424) specified during OTA client configuration at section 4.1 Configure and Build the Original Zigbee OTA Client Image.


bootloader-storage-internal-single.slcp
readme.md
Bootloader Storage Slot Setup
Pin Tool
View Source

Bootloader Storage Slot Setup

Bootloader Storage Slot Setup

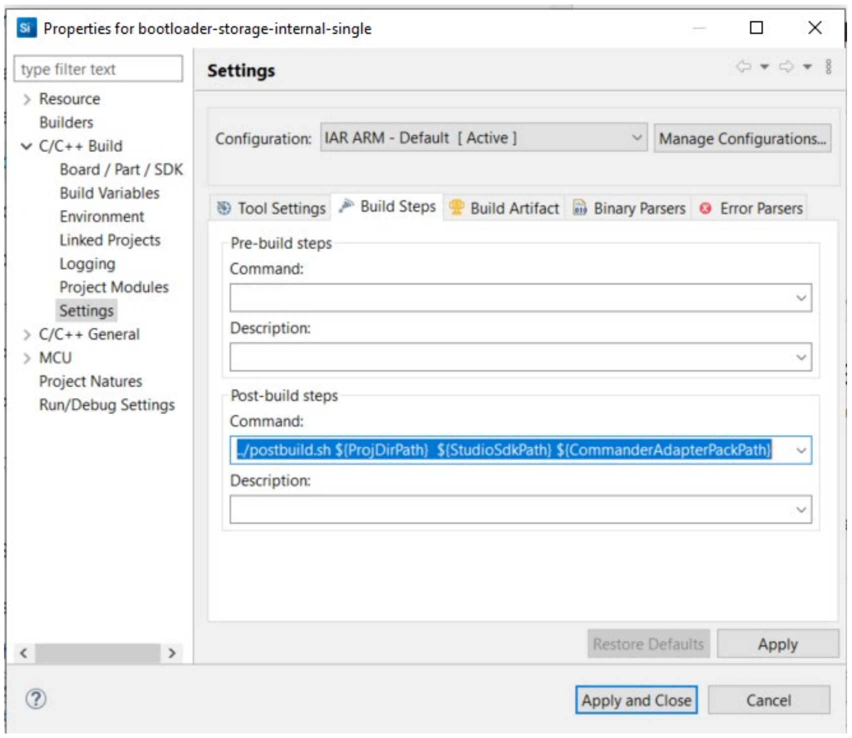
Enable Slot 0

Start Address: 540672

Slot Size: 458752

Enable Slot 1

3. For the build to generate the required combined bootloader image, right-click on the project name in the Project Explorer view and select **Properties**. Then add this string `../postbuild.sh ${ProjDirPath} ${StudioSdkPath} ${CommanderAdapterPackPath}` to the command box under **Post-build steps** as shown below. Click **Apply and Close**.



Configuration: IAR ARM - Default [Active]

Post-build steps
Command: ../postbuild.sh ${ProjDirPath} ${StudioSdkPath} ${CommanderAdapterPackPath}

4. Build the bootloader.
5. The `bootloader-storage-internal-single-combined.s37` bootloader image is generated. Note that only the Series 1 device requires combined bootloader. See *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* for more information.

4.6 Load the Original Client Image

Flash the appropriate bootloader for your board. In this case, you can use the same bootloader for both the OTA Client and Server if you are using the same board (for example, BRD4161A). You can use Simplicity Studio to do this (see *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher* for instructions on loading application and bootloader firmware) or use Simplicity Commander commands (see *UG162: Simplicity Commander Reference Guide* for additional detail):

```
commander flash -d EFR32 [path to bootloader-storage-internal-single-combined.s37]
commander flash -d EFR32 [path to ZNet_OTA_Client_7.gbl]
```

4.7 Load the EZSP NCP Server Software

To use an EFR32MG device connected to a PC host through USB, it is necessary to build and load the EZSP NCP software onto the target chip. The following describes the process of locating and flashing a prebuilt NCP image to an EFR32MG12 as an example for the NCP target chip. You can also build your own images as described in *AN1320: Building a Customized NCP Application with Zigbee EmberZNet 7.x* if the sample image cannot be located.

1. Locate the prebuilt sample ncp-uart-hw from GSDK 4.x release. As an example, the following is the image location for the **BRD4161A**.

   `C:\SiliconLabs\SimplicityStudio\v5\offline\com.silabs.sdk.stack.super_`**`<GSDK_Ver>`**`\protocol\zigbee\demos\ncp-uart-hw\ ncp-uart-hw-brd4161a.s37`

2. Locate the prebuilt `bootloader-storage-internal-single-combined.s37`.

3. Flash in the bootloader image followed by the ncp-uart-hw image as shown below.

```
commander flash -d EFR32 [path to bootloader-storage-internal-single-combined.s37]
commander flash -d EFR32 [path to ncp-uart-hw-brd4161a.s37]
```

4.8 Load the Updated OTA Client Image on the Server

The OTA Server application looks for OTA files in a directory called `ota-files` that is a sub-directory in the current working directory from which the server application is launched.

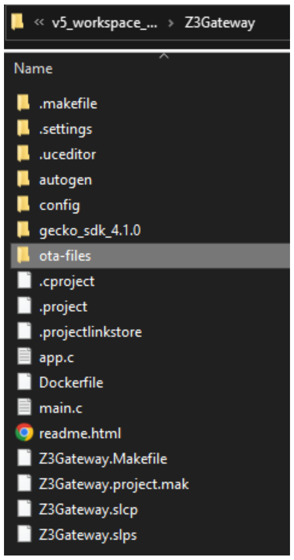
For Linux

1. Launch a Linux shell on the OTA Server PC.
2. Locate the previously built OTA file `ZNet_OTA_Client_7_v22.ota`.
3. Navigate to the directory where the OTA Server is located (e.g. `/home/<`**`UserID`**`>/Z3Gateway`)
4. Make a subdirectory and copy the OTA file to it.

```
cd /home/<UserID>/Z3Gateway
mkdir ota-files
cp -i [PATH To ZNet_OTA_Client_7_v22.ota] ota-files/
```

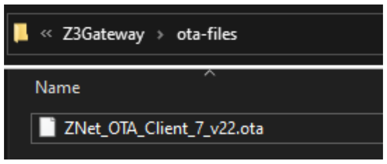
For Windows Docker Container

1. Locate the previously built OTA file `ZNet_OTA_Client_7_v22.ota`.
2. Use Windows Explorer to create a directory called `ota-files` under the directory where the Z3Gateway OTA Server application will be built and run using Docker container.



.makefile
.settings
.uceditor
autogen
config
gecko_sdk_4.1.0
ota-files
.cproject
.project
.projectlinkstore
app.c
Dockerfile
main.c
readme.html
Z3Gateway.Makefile
Z3Gateway.project.mak
Z3Gateway.slcp
Z3Gateway.slps

3. Copy the OTA file to the `ota-files` directory.



ZNet_OTA_Client_7_v22.ota

4. Make sure to rerun docker build to update the OTA file in the container.

4.9 Run the Zigbee OTA Client / Server

1. Make sure the WSTK for the NCP device is connected to the PC with a USB cable.
2. Launch the OTA Server by executing the OTA server application from the command line and pass it the location of the communications port. The communications port will either be a USB or serial port device. The method for determining which port the EFR32MG12 is connected to depends on the operating system of the host PC.

Under Linux

The USB port is usually registered as /dev/ttyUSBX, where X is a number as assigned by the operating system. For example, the following would use `ttyUSB0`.

a) Launch a Linux Shell.

b) Change to the directory where the OTA Server application image is located. For example: `cd /home/`**`<UserID>`**`/Z3Gateway/build/debug/`

c) Start the server by running `./Z3Gateway -n 0 -p /dev/ttyUSB0`.

Under Windows Running Docker Container

Make sure you have followed the instructions from the previously mentioned *AN1389: Running Zigbee Host Applications in a Docker Container* to build and run the Docker container for the Z3Gateway host application.

The USB port is either COM1, COM2, COM3, and so on. The COM port can be determined by watching the Ports (COM & LPT) section of Windows Device Manager while unplugging and replugging the WSTK's USB connector. The following example uses COM1.

a) Start a command window as Administrator.

```
silink.exe -automap 4900
```

b) Run the following silink command to start a silink session.

c) Start a new command window.

d) Navigate to the directory where the Z3Gateway host app is built.

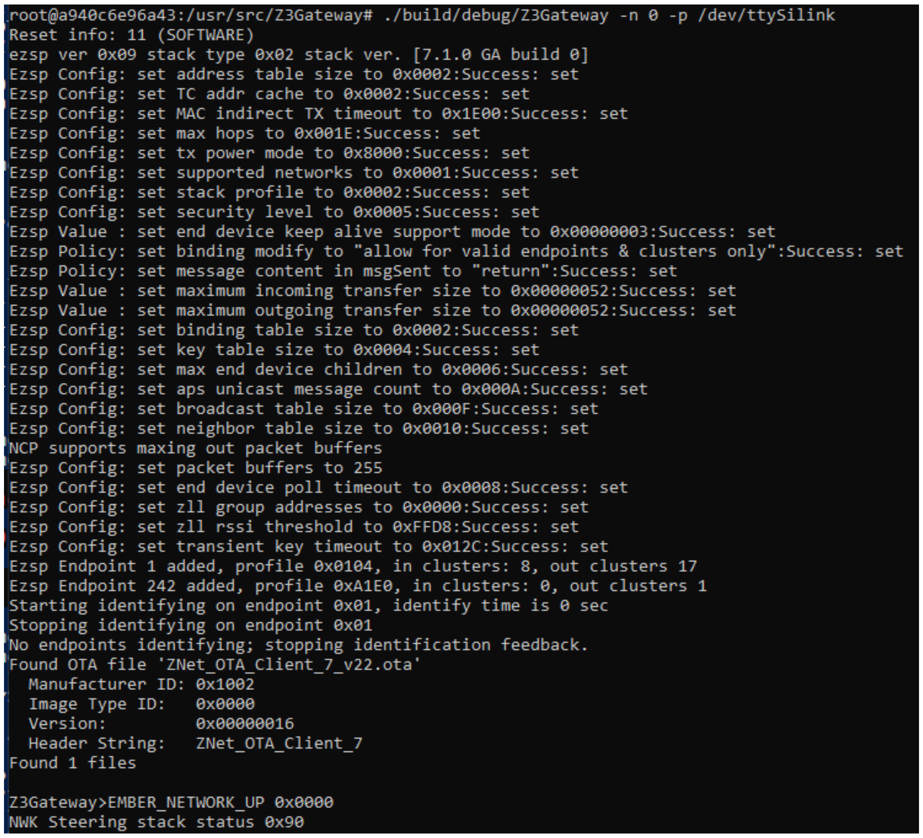
e) Assuming a Dockerfile has already been created based on *AN1389*, run the following commands in the command window to start the container.

```
docker build . -t z3gateway
docker run -it z3gateway
```

f) To run your host app type the command within the container.

```
./build/debug/Z3Gateway -n 0 -p /dev/ttySilink
```

g) The following shows a successful launch of the Z3Gateway host application from the container.



```
root@a940c6e96a43:/usr/src/Z3Gateway# ./build/debug/Z3Gateway -n 0 -p /dev/ttySilink
Reset info: 11 (SOFTWARE)
ezsp ver 0x09 stack type 0x02 stack ver. [7.1.0 GA build 0]
Ezsp Config: set address table size to 0x0002:Success: set
Ezsp Config: set TC addr cache to 0x0002:Success: set
Ezsp Config: set MAC indirect TX timeout to 0x1E00:Success: set
Ezsp Config: set max hops to 0x001E:Success: set
Ezsp Config: set tx power mode to 0x8000:Success: set
Ezsp Config: set supported networks to 0x0001:Success: set
Ezsp Config: set stack profile to 0x0002:Success: set
Ezsp Config: set security level to 0x0005:Success: set
Ezsp Value : set end device keep alive support mode to 0x00000003:Success: set
Ezsp Policy: set binding modify to "allow for valid endpoints & clusters only":Success: set
Ezsp Policy: set message content in msgSent to "return":Success: set
Ezsp Value : set maximum incoming transfer size to 0x00000052:Success: set
Ezsp Value : set maximum outgoing transfer size to 0x00000052:Success: set
Ezsp Config: set binding table size to 0x0002:Success: set
Ezsp Config: set key table size to 0x0004:Success: set
Ezsp Config: set max end device children to 0x0006:Success: set
Ezsp Config: set aps unicast message count to 0x000A:Success: set
Ezsp Config: set broadcast table size to 0x000F:Success: set
Ezsp Config: set neighbor table size to 0x0010:Success: set
NCP supports maxing out packet buffers
Ezsp Config: set packet buffers to 255
Ezsp Config: set end device poll timeout to 0x0008:Success: set
Ezsp Config: set zll group addresses to 0x0000:Success: set
Ezsp Config: set zll rssi threshold to 0xFFD8:Success: set
Ezsp Config: set transient key timeout to 0x012C:Success: set
Ezsp Endpoint 1 added, profile 0x0104, in clusters: 8, out clusters 17
Ezsp Endpoint 242 added, profile 0xA1E0, in clusters: 0, out clusters 1
Starting identifying on endpoint 0x01, identify time is 0 sec
Stopping identifying on endpoint 0x01
No endpoints identifying; stopping identification feedback.
Found OTA file 'ZNet_OTA_Client_7_v22.ota'
  Manufacturer ID: 0x1002
  Image Type ID:   0x0000
  Version:         0x00000016
  Header String:   ZNet_OTA_Client_7
Found 1 files

Z3Gateway>EMBER_NETWORK_UP 0x0000
NWK Steering stack status 0x90
```

3. Connect to the OTA Client device in your Simplicity Studio Serial 1 window or by launching a Windows command prompt and connecting via telnet to its IP address on port 4901.
4. Make sure both the client and server devices have Over-the-Air Bootloading Cluster printing enabled by running the following commands:
   a) On the OTA Client: `plugin ota-storage-common printImages`
   b) On the OTA Server: `plugin ota-storage-common printImages`

   In both cases, output should print to the CLI. If no output is visible, go back and verify the configuration of the application and its Debug Printing configuration.
5. Make sure that the client is running the older version of the client software. From the OTA client's CLI type the following command. You should see the corresponding output:

```
Z3Light>plugin ota-client info
Client image query info
Manuf ID:      0x1002
Image Type ID: 0x0000
Current Version: 0x00000001
Hardware Version: NA
Query Delay ms:            300000
Server Discovery Delay ms: 600000
Download Delay ms:         0
Run Upgrade Delay ms:      600000
Verify Delay ms:           10
Download Error Threshold:  10
Upgrade Wait Threshold:    10
```

   Note that the current version field says 0x00000001.

6. Verify that the OTA Server has newer image in its OTA storage. From the OTA server's CLI type the following command. You should see the corresponding output:

```
plugin ota-storage-common printImages
Z3Gateway>Image 0
  Header Version: 0x0100
  Header Length:  56 bytes
  Field Control:  0x0000
  Manuf ID:       0x1002
  Image Type:     0x0000
  Version:        0x00000016
  Zigbee Version: 0x0002
  Header String:  ZNet_OTA_Client_7
  Image Size:     331502 bytes
  Total Tags: 1
    Tag: 0x0000
      Length: 331440

1 images in OTA storage.
```

   Note that the version number is 0x00000016. The exact size of the image may vary.

7. Erase any previous version file that has been saved in the client. This is not normally necessary in a production system, but in this example a previous version may cause problems. Enter the following command on the client:

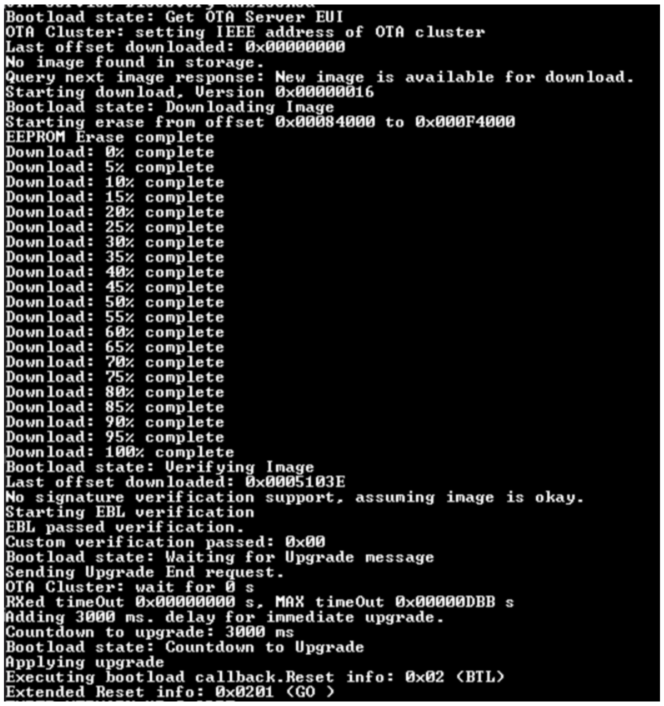
```
plugin ota-storage-common delete 0.
```

8. Turn off receive message printing to limit console output during the download. Skipping this step will not negatively impact the functionality.
   a) On the server: `option print-rx-msgs disable`
   b) On the client: `option print-rx-msgs disable`
9. Join the devices into the same network.
   a) On the server: `net leave`
   b) On the server: `plugin network-creator start 1`
   c) On the server: `plugin network-creator-security open-network`
   d) On the client: `plugin network-steering start 0`

   If the client joins are successful, you should see a dialog about trust center interactions in the output.
10. To verify that the client is connected to the server, type `info` on both the client and the server. The PAN IDs should be the same.
11. Start the OTA Client's state machine on the client: `plugin ota-client start`

    Note: The download may take up to 10 minutes.

12. If all goes well, you should see the following output:



```
Bootload state: Get OTA Server EUI
OTA Cluster: setting IEEE address of OTA cluster
Last offset downloaded: 0x00000000
No image found in storage.
Query next image response: New image is available for download.
Starting download, Version 0x00000016
Bootload state: Downloading Image
Starting erase from offset 0x00084000 to 0x000F4000
EEPROM Erase complete
Download: 0% complete
Download: 5% complete
Download: 10% complete
Download: 15% complete
Download: 20% complete
Download: 25% complete
Download: 30% complete
Download: 35% complete
Download: 40% complete
Download: 45% complete
Download: 50% complete
Download: 55% complete
Download: 60% complete
Download: 65% complete
Download: 70% complete
Download: 75% complete
Download: 80% complete
Download: 85% complete
Download: 90% complete
Download: 95% complete
Download: 100% complete
Bootload state: Verifying Image
Last offset downloaded: 0x0005103E
No signature verification support, assuming image is okay.
Starting EBL verification
EBL passed verification.
Custom verification passed: 0x00
Bootload state: Waiting for Upgrade message
Sending Upgrade End request.
OTA Cluster: wait for 0 s
RXed timeOut 0x00000000 s, MAX timeOut 0x00000DBB s
Adding 3000 ms. delay for immediate upgrade.
Countdown to upgrade: 3000 ms
Bootload state: Countdown to Upgrade
Applying upgrade
Executing bootload callback.Reset info: 0x02 (BTL)
Extended Reset info: 0x0201 (GO )
```

4.10 Repeating the Procedure

Once a Zigbee OTA client is upgraded through the Zigbee over-the-air bootload cluster, it can only be upgraded again if the server has a different version of software. Therefore, to perform a Zigbee OTA firmware download again, it is necessary to do one of the following:

Downgrade the Running Image to the Original Version

Re-run the steps in the section 4.6 Load the Original Client Image to load the original version 1 of the OTA Client software image.

Create Another Image with a Newer Software Version

Run the following steps:

1. Re-run the steps in section 4.2 Configure and Build the Updated Zigbee OTA Client Image but specify a different Firmware Version that is greater than 22.
2. Re-run the steps in section 4.8 Load the Updated OTA Client Image on the Server and replace the older OTA Client upgrade image with the new one.

Make sure that the **--version** argument passed to Image-builder reflects the newest version number.

5 Advanced Topics

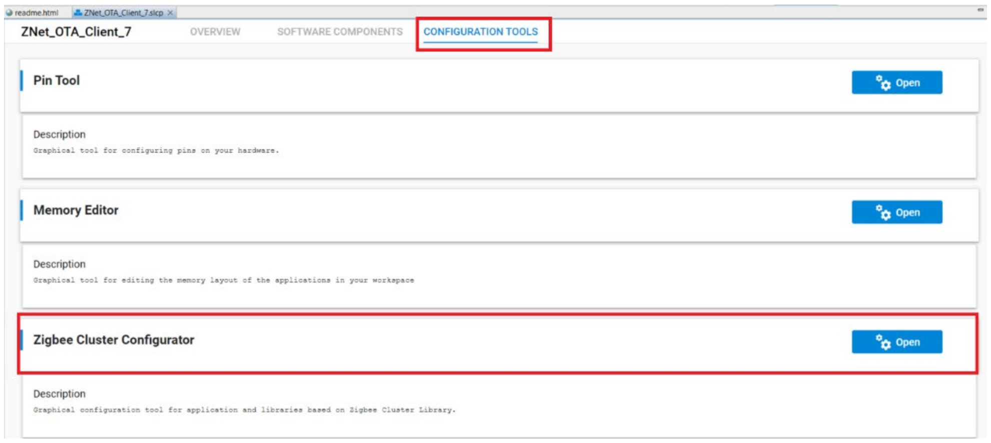
This section presents several advanced topics that allow the developer to customize the bootload to their specific hardware and software.

5.1 Specifying Your Own Manufacturer ID

Follow these steps to specify a manufacturer ID other than 0x1002, which is the Silicon Labs-specific manufacturing ID.

Each member of the Connectivity Standards Alliance (CSA) has its own manufacturer ID. The list of manufacturer IDs is maintained in CSA Document 05-3874 Manufacturer Code Database. If your company does not have a manufacturer ID, then it must request one from the CSA (https://csa-iot.org/).

1. When configuring the OTA Client in the **Project Configurator**, do the following:
   a) Select the **CONFIGURATION TOOLS** tab and open the **Zigbee Cluster Configurator**.

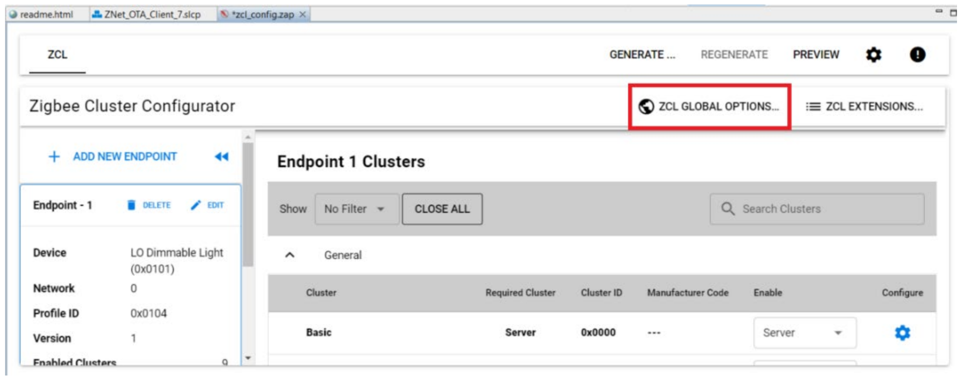


Description
Graphical tool for configuring pins on your hardware.

Description
Graphical tool for editing the memory layout of the applications in your workspace

Description
Graphical configuration tool for application and libraries based on Zigbee Cluster Library.

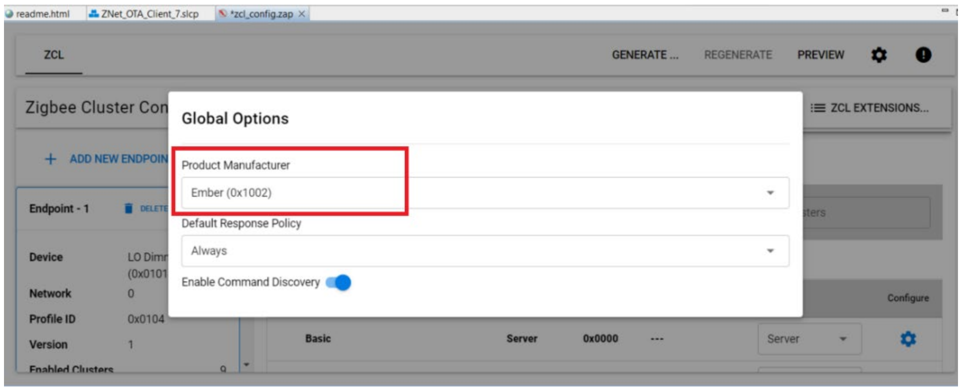
   b) Click on **ZCL GLOBAL OPTIONS**.



Device	LO Dimmable Light (0x0101)
Network	0
Profile ID	0x0104
Version	1
Enabled Clusters	9

Cluster	Required Cluster	Cluster ID	Manufacturer Code	Enable	Configure
Basic	Server	0x0000	---	Server	

   c) In the **Product Manufacturer** text box entry, select or enter your own manufacturer code.



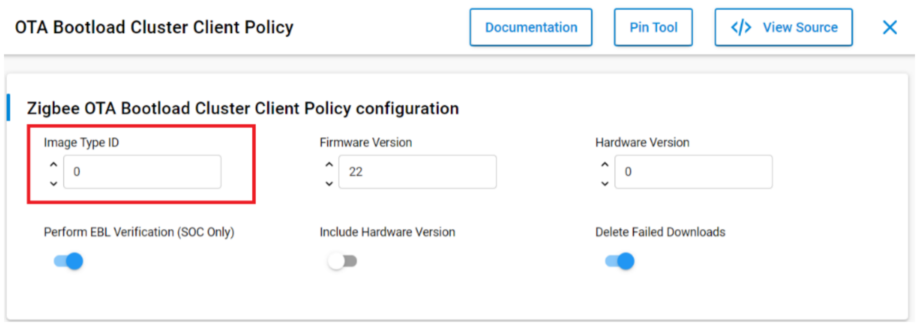
Product Manufacturer
Ember (0x1002)
Default Response Policy
Always
Enable Command Discovery

   d) Save the change with shortcut key CTRL-S or File->Save.
   e) Re-compile the application.
2. When generating the Zigbee OTA image file, do the following.
   a) Specify your own manufacturer ID by passing that value to the **--manuf-id** argument when executing the image-builder command.

Note: You do NOT need to modify the manufacturer ID of the OTA server in this example. The server may have a different manufacturer ID than the OTA Client. You may choose to modify the manufacturer ID to match the OTA server device that your company is building.

5.2 Specifying Your Own Image Type ID

Follow these steps to specify an image type ID other than 0x0000.
1. When configuring the OTA Client in Project Configurator, do the following:
   a) Select the **OTA Bootload Cluster Client Policy** software component.
   b) In the **Image Type ID** text entry box, enter your own image type ID code.


OTA Bootload Cluster Client Policy

Zigbee OTA Bootload Cluster Client Policy configuration
Image Type ID: 0
Firmware Version: 22
Hardware Version: 0
Perform EBL Verification (SOC Only): on
Include Hardware Version: off
Delete Failed Downloads: on

   c) Re-compile the application using IAR workbench.
2. If the project does not generate OTA file automatically, the new **Image Type** argument will need to be specified on the command line when running the Image-builder with --image-type.

5.3 Modifying the OTA Client to Use a Different EEPROM

Many different external storage parts are available, and the choice of which one is used is based on many different factors. *AN772: Using the Ember Application Bootloader* and *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher* detail several different parts supported by Silicon Labs and their various device parameters.

If the external storage part does not support read-modify-write, then the Project Configurator configuration must be updated. The following describes the process:
1. Open the OTA client's Project Configurator.
2. Select the **SOFTWARE COMPONENT** tab.
3. Select the **OTA Simple Storage EEPROM Driver** plugin and disable the option labeled **EEPROM Read-modify-write device Support**.
4. Re-compile the application.

Note: A new Bootloader will be required to support the change in EEPROM. You must load a new bootloader by following the steps in section 4.6 Load the Original Client Image, and selecting one appropriate to the bootloader you are using.

5.4 Using a Cross-Compiler for the OTA Server

It is possible to use a cross-compiler for building the OTA server for another target system. For example, building on an x86 Windows PC for an ARM Linux system. This process is relatively straight forward once the cross-compiler is properly installed and configured. To do this you must run make and pass it the name/location of the compiler and linker on the command-line.

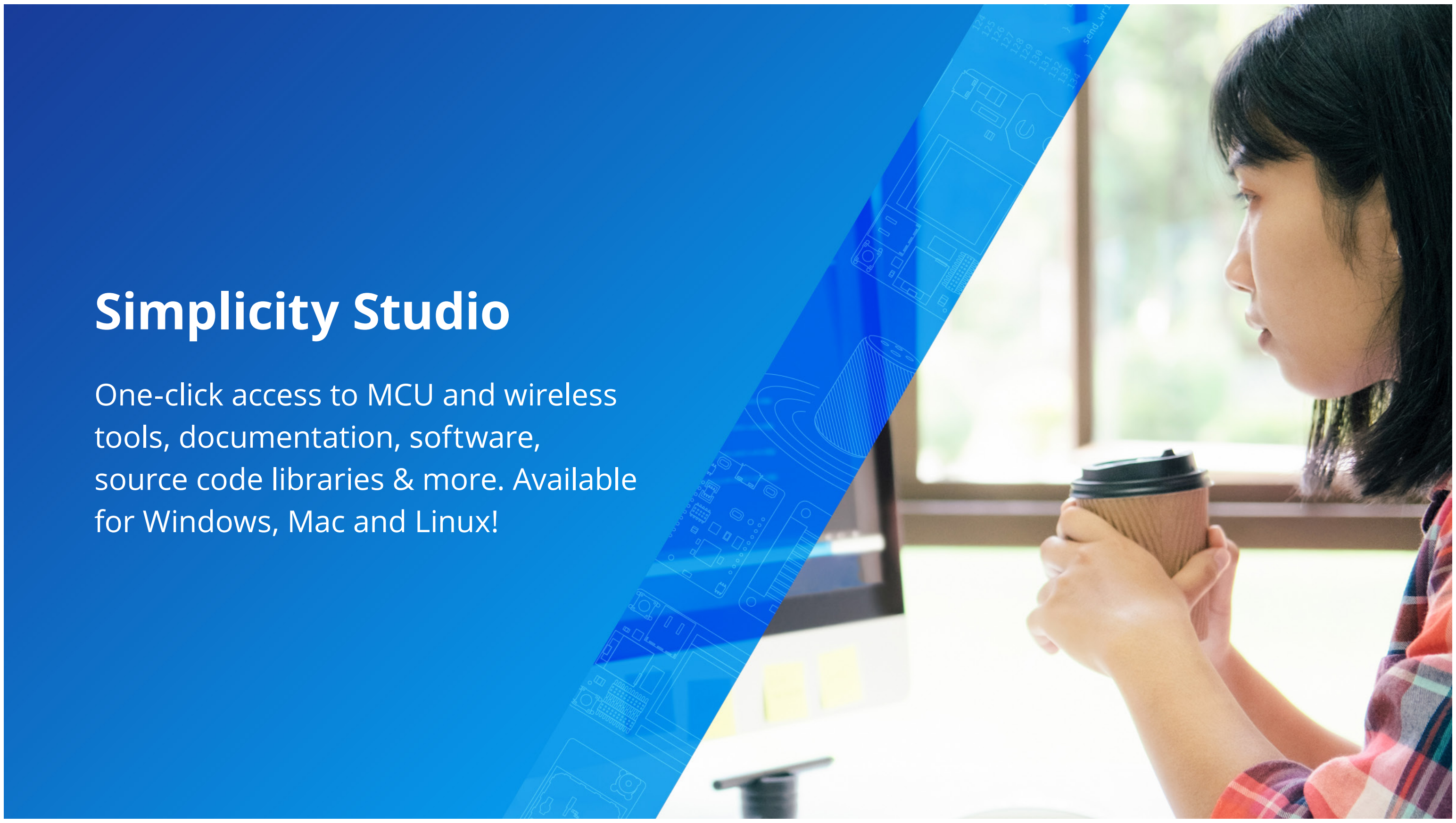
For example:

```
make CC=/opt/crosstool/gcc-4.0.1-glibc-2.3.5/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu-gcc
LD=/opt/crosstool/gcc-4.0.1-glibc-2.3.5/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu-gcc
```

The Makefile expects a GCC style command-line compiler, and it uses the compiler command (rather than the linker command directly) for linking.

5.5 Using Encrypted GBL Images

Encrypted images protect the new application image from inspection or tampering by using a symmetric encryption key. For more information about the Gecko Bootloader and encrypted GBL images, including changing the bootloader type and loading an encryption key onto the device, see *UG489: Silicon Labs Gecko Bootloader User's Guide for GSDK 4.0 and Higher*.


Simplicity Studio
One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com