QSG102: Thread Border Router Add-On Kit Quick-Start Guide

The Silicon Labs Thread Border Router Add-On Kit (RD-0004-0201) adds a border router demonstration system to the EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000B).

KEY POINTS

- Set Up a EFR32 Wireless Starter Kit
- Set Up a Raspberry Pi
- Install Simplicity Studio
- Install Border Router Software
- Install NCP and End Device Firmware
- Install Commissioner Software
- Demonstrate the Thread Border Router
- Resources

Because most users have found it more convenient to acquire the Border Router Raspberry Pi hardware from third-party sources, as of Thread Border Router version 3.1.0, the kit RD-0004-0201 is being phased out and will no longer available for purchase from Silicon Labs. Throughout this document the part number RD-0004-0201 is used to refer to either the kit acquired previously from Silicon Labs or built with components from a third-party.

This document describes the Thread Border Router Kit, how to acquire and set up the hardware, how to install the required software, and how to demonstrate the Thread Border Router.

An EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000B) provides a necessary component for the Border Router and is required to obtain a software registration and support key. The SLWSTK6000B kit contains three wireless starter kit mainboards with EFR32MG12 chipsets. One of the mainboards is required for a Thread network co-processor (NCP), and one or more of the mainboards may be used as Thread end nodes. The following reference designs, available from Silicon Labs, may also be used as end nodes:

- RD-0039-0201 Capacitive Sense Dimmer Switch Reference Design
- RD-0030-0201 Contact Sensor Reference Design
- RD-0100-0201 Smart Outlet Reference Design
- RD-0099-0201 Occupancy Sensor Reference Design
- RD-0035-0601 Lighting Reference Design
- RD-0098-0401 Lighting Reference Design

This document refers to the Thread Border Router version 1.3.0, which requires Silicon Labs Thread stack version 2.6.0 GA, or later.

There are two other important changes for version 1.3.0 compared with previous versions:

The kit RD-0004-0201 shipped from Silicon Labs used a CEL USB dongle as the NCP. To improve ease-of-use, the CEL USB dongle is replaced with one of the three wireless starter kit mainboards available with the EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000B).

The EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000B) with EFR32MG12 chipsets replaces the EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000A) with EFR32MG1 chipsets. Previous users of the SLWSTK6000A version will need to upgrade to the SLWSTK6000B version.

1. Introduction

A border router is an essential component of a Thread network that manages the traffic between the Thread network and adjacent IP networks. A border router facilitates a number of key capabilities including the following:

- Commissioning of Thread nodes
- GUA (Global Unicast Address) or ULA (Unique Local Address) assignment
- IP routing between the Thread network and adjacent IP networks

The Silicon Labs Thread Border Router Add-On Kit (RD-0004-0201) adds a border router demonstration system to the EFR32 Mighty Gecko Wireless SoC Starter Kit (SLWSTK6000B). This document describes:

- Setting up the Thread Border Router Hardware
- Setting up the Thread End Node Hardware
- Installing Simplicity Studio
- Installing the Border Router Software
- Installing the Thread End Device Software
- Installing the Android or iOS Device Software
- Demonstrating the Thread Border Router
- Accessing other resources

This version of the Thread Border Router offers:

- Important bug fixes
- Support for EFR32 Mighty Gecko Wireless SoC Starter Kit
- Support for the Thread Group commissioning app for Android or iOS
- Support for package manager installation
- Support for NAT64 and DNS64 IPv6 transition technologies
- Automatic Prefix Delegation
- Run-time control over network parameters
- DNS lookup
- zclip-cli utility for device interaction
- zclip-rd Resource Directory

Typical Thread Border Router setups include:

- Thread Border Router with Wi-Fi Access Point
- Thread End Devices such as EFR32 Mighty Gecko Wireless SoC Starter Kit
- Host computer with Simplicity Studio and SSH client for accessing the Border Router
- Commissioning device such as an Android or iOS handset
- IPv6 Ethernet Connectivity

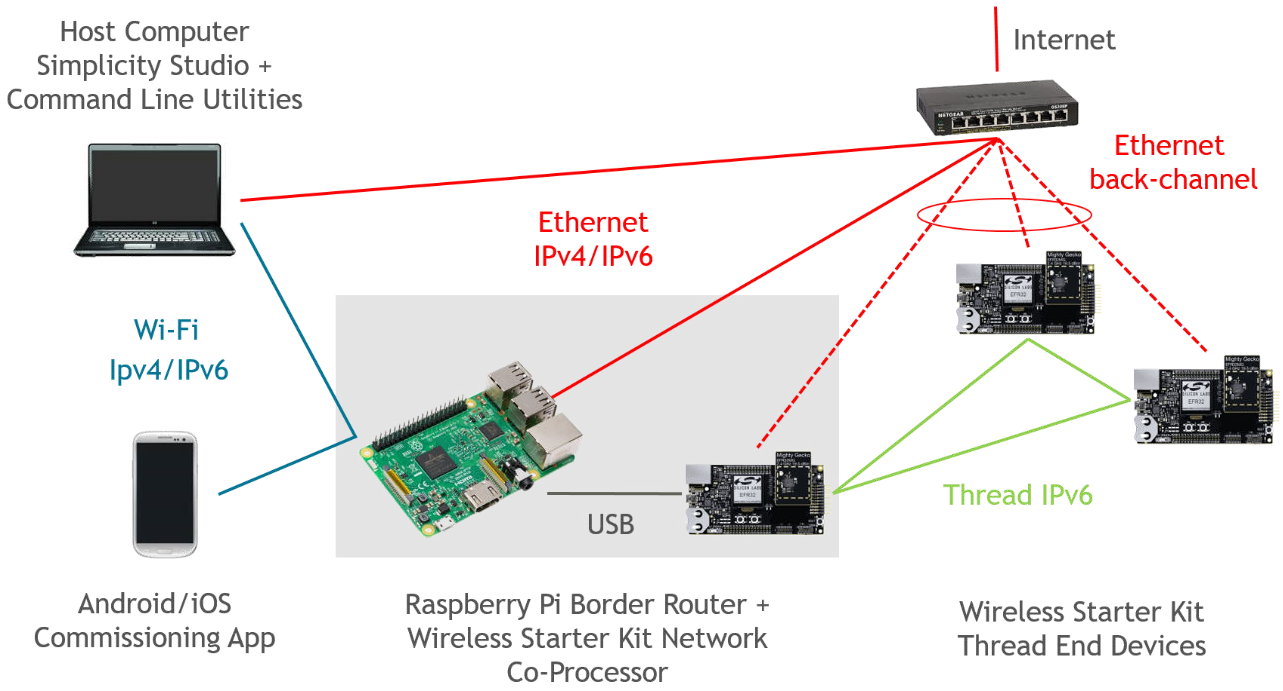

Figure 1.1. Thread System Components

2. Order, Register, and Set Up the EFR32 Mighty Gecko Wireless SoC Starter Kit

2.1 Order the EFR32 Mighty Gecko Wireless SoC Starter Kit

Kits may be ordered from http://www.silabs.com/products/wireless/mesh-networking/zigbee/Pages/zigbee.aspx.

The EFR32 Mighty Gecko Wireless SoC Starter Kit includes the following:

- 3 x Wireless starter kit mainboard
- 3 x EFR32MG12 2.4 GHz 19.5 dBm radio board
- 3 x EFR32MG12 2.4 GHz 10 dBm radio board
- AA Battery Board (supports running +19.5 dBm from battery)
- Cables
- EFR32MG Getting Started Card

QSG113: Getting Started with Silicon Labs Thread describes how to set up and start using the Wireless SoC starter kit with the Silicon Labs Thread SDK.

2.2 Register on the Support Portal

Registration allows access to the support portal, and is a necessary step to access required software and documentation. The instructions that come with the EFR32 Mighty Gecko Wireless SoC Starter Kit include a unique registration key and explain the registration procedure.

2.3 Set Up Hardware

1. Connect each wireless starter kit mainboard and the host computer to an Ethernet switch with an Ethernet cable as shown in Figure 1.1 Thread System Components on page 3. These connections will permit programming and network analysis of the NCP and end devices. Optionally, end devices may be connected to the host computer by USB rather than Ethernet.
2. Refer to *QSG113: Getting Started with Silicon Labs Thread* for instructions on setting up and using the EFR32 Mighty Gecko Wireless SoC Starter Kit.

3. Order and Set Up a Raspberry Pi

3.1 Order a Raspberry Pi

Refer to https://www.raspberrypi.org for recommended vendors for each component.

- Raspberry Pi 2 Model B or Raspberry Pi 3 Model B
- 16 GB MicroSD card
- Edimax EW-7811UN USB 2.0 Wireless Adapter (required for Raspberry Pi 2 Model B only)
- Power supply

3.2 Set Up the Raspberry Pi

1. Connect the USB Wi-Fi Adapter to one of the Raspberry Pi's USB ports.
2. Connect the Raspberry Pi's Ethernet port to the Internet with an Ethernet cable.

   Note that it may be desirable to use an Ethernet switch as shown in Figure 1.1 Thread System Components on page 3, however, it is not necessary that the Raspberry Pi be on the same Ethernet network as the host computer and Wireless Starter Kit mainboards.

3. Connect one of the Wireless Starter Kit mainboards to the Raspberry Pi with a USB cable. This will become the network co-processor as shown in Figure 1.1 Thread System Components on page 3.
4. Connect a monitor to the HDMI port and a keyboard to a free USB port.
5. Plug in the Thread Border Router power supply.

The Raspberry Pi and NCP Wireless Starter Kit are collectively referred to as the Border Router.

4. Install Simplicity Studio on the Host Computer

Simplicity Studio is required to flash Thread end node firmware. Refer to *QSG113: Getting Started with Silicon Labs Thread*, for a detailed tutorial. The tutorial includes installation and configuration instructions, and describes the process for compiling and running an example for the EM35x or EFR32MG devices. Simplicity Studio is required for the steps in section 8. Demonstrate the Thread Border Router.

5. Install the Border Router Software on the Raspberry Pi

The Border Router Debian package includes the border-router-mgmt-app, commission-proxy-app, ip-driver-app, node.js demonstration application, and all required configuration, drivers and support packages.

The border-router-mgmt-app, commission-proxy-app, and ip-driver-app are available with the Thread stack distribution and may be inspected and built using Simplicity Studio. See *UG116: Developing Custom Border Router Applications* for instructions for building these applications. The node.js demonstration application source is distributed with the Border Router Debian package. The configuration, drivers and support packages are also described in See *UG116: Developing Custom Border Router Applications.*

1. Use a host computer to install the Raspbian Stretch Lite operating system on the SD card as described here: https://www.raspberrypi.org/downloads/raspbian/ (currently tested and supported release is Stretch Lite 2017-11-29, this archived version can be download here: http://downloads.raspberrypi.org/raspbian_lite/images/raspbian_lite-2017-12-01/).
2. Install the SD card in Raspberry Pi and power it on.
3. Log in and configure the keyboard layout.

   The default username is “pi” and password is “raspberry.” On first reboot you are prompted to change the password. Configure the default keyboard layout and optionally enable the SSH server with:

```
$ sudo raspi-config
```

4. Configure the keys and install the silabs-border-router package.

```
$ sudo apt-get update
$ sudo apt-get install dirmngr
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 90CE4F77
$ sudo chmod 666 /etc/apt/sources.list
$ sudo echo deb http://devtools.silabs.com/solutions/apt stretch main >> /etc/apt/sources.list
$ sudo apt-get update
$ sudo apt-get install silabs-border-router
$ sudo reboot
```

Note: In the following procedures you will power the Thread Border Router on and off by connecting and disconnecting its power supply. The red power LED will illuminate on the Border Router and the green activity LED will blink until the boot process has completed. The proper procedure for power down is to issue the following command before disconnecting power.

```
$ sudo shutdown -h now
```

6. Install NCP and End Device Firmware on the EFR32 Wireless Starter Kit Mainboards

The NCP and end device firmware is programmed on the wireless starter kit main boards using Simplicity Studio. One mainboard is required for the NCP, and one or more mainboards are required for end devices. NCP and end device firmware is available in the Thread stack distribution and may be inspected and built using Simplicity Studio. For convenience, pre-compiled NCP and end device firmware files are distributed with the Border Router file system, as is the firmware and bootloaders for the following reference designs:

- RD-0039-0201 Capacitive Sense Dimmer Switch Reference Design
- RD-0030-0201 Contact Sensor Reference Design
- RD-0100-0201 Smart Outlet Reference Design
- RD-0099-0201 Occupancy Sensor Reference Design
- RD-0035-0601 Lighting Reference Design
- RD-0098-0401 Lighting Reference Design

These files may be transferred from the Border Router to the host computer with a utility such as WinSCP or SCP as described in Section 7.5 of *UG116: Developing Custom Border Router Applications*.

Note: End device reference designs are only supported in the Silicon Labs Thread stack up to version 2.6.

6.1 Install Firmware on the NCP

Option 1: Transfer the pre-compiled bootloaders and firmware from the Raspberry Pi to the host computer.

Note the back slashes "\" in the paths denote line breaks and should not be copied.

Type	EFR32MG12
First Stage Bootloader	/opt/siliconlabs/threadborderrouter/firmware/ncp-uart-hw/efr32mg12p432f1024gl125/\ first_stage_btl_efx32xg12.s37
Bootloader	/opt/siliconlabs/threadborderrouter/firmware/ncp-uart-hw/efr32mg12p432f1024gl125/\ bootloader-uart-xmodem-efr32mg12p432f1024gl125.s37
Programming File	/opt/siliconlabs/threadborderrouter/firmware/ncp-uart-hw/efr32mg12p432f1024gl125/\ ncp-uart- efr32mg12p432f1024gl125.s37

Option 2: Build the NCP sample application with Simplicity Studio on the host computer.

Note the back slashes "\" in the paths denote line breaks and should not be copied.

Type	EFR32MG12
First Stage Bootloader (pre-compiled)	/SimplicityStudio_v4/developer/sdks/gecko_sdk_suite/v2.0/\ platform/bootloader/build/first_stage/iarfirst_stage_btl_efx32xg12.s37
Bootloader (pre-compiled)	/SimplicityStudio_v4/developer/sdks/gecko_sdk_suite/v2.0/\ platform/bootloader/sample-apps/bootloader-uart-xmodem/efr32mg12p432f1024gl125/\ bootloader-uart-xmodem-efr32mg12p432f1024gl125.s37
Bootloader (source + libraries)	Build sample application: Gecko Bootloader >> UART XMODEM Bootloader for board BRD 4161 and part efr32mg12p432f1024gl125.
Programming File (source + libraries)	Build sample application: Thread Stack >> NCP UART HW (Hardware Flow Control) for board BRD4161 and part efr32mg12p432f1024gl125.

Connect one of three the EFR32 Wireless Starter Kit Mainboards to the host computer with Simplicity Studio installed and see *QSG113: Getting Started with Silicon Labs Thread* for programming instructions.

IMPORTANT: The chip should be erased before uploading image. This ensures any old network settings will be erased.

6.2 Install Firmware on the End Devices

Option 1: Transfer the pre-compiled bootloaders and firmware from the Raspberry Pi to the host computer.

Note the back slashes "\" in the paths denote line breaks and should not be copied.

Type	EFR32MG121
First Stage Bootloader	/opt/siliconlabs/threadborderrouter/firmware/sensor-actuator/\ efr32mg12p432f1024gl125\first_stage_btl_efx32xg12.s37
Bootloader	/opt/siliconlabs/threadborderrouter/firmware/sensor-actuator/\ efr32mg12p432f1024gl125\bootloader-storage-internal-efr32mg12p432f1024gl125.s37
Programming File	/opt/siliconlabs/threadborderrouter/firmware/sensor-actuator/\ efr32mg12p432f1024gl125\sensor-actuator- efr32mg12p432f1024gl125.s37

Option 2: Build the end device sample application with Simplicity Studio on the host computer.

Note the back slashes "\" in the paths denote line breaks and should not be copied.

Type	EFR32MG12
First Stage Bootloader (pre-compiled)	/SimplicityStudio_v4/developer/sdks/gecko_sdk_suite/v2.0/\ platform/bootloader/build/first_stage/iar/first_stage_btl_efx32xg12.s37
Bootloader (source + libraries)	Build sample application: Gecko Bootloader >> Internal Storage Bootloader (Multiple Images) for board BRD4161 and part efr32mg12p432f1024gl125.
Programming File (source + libraries)	Build sample application: Thread Stack >> Sensor/Actuator for board BRD4161 and part efr32mg12p432f1024gl125.

Connect one of three the EFR32 Wireless Starter Kit Mainboards to the host computer with Simplicity Studio installed and see *QSG113: Getting Started with Silicon Labs Thread* for programming instructions.

IMPORTANT: The chip should be erased before uploading image. This ensures any old network settings will be erased.

7. Install the Commissioning Application on an Android or iOS Device

An Android or iOS handset or tablet with the Thread Group commissioning app is required to commission Thread end devices. The Thread Group commissioning app is distributed in the Google Play store and is also distributed in the Thread Border Router file system. The file may be transferred from the Border Router to a host with a utility such as WinSCP or SCP as described in *UG116: Developing Custom Border Router Applications*. To access the source for the commissioning app, refer to the Thread Group Bitbucket repository https://bitbucket.org/threadgroup/, and request access to the repository through the Thread Group.

Device	Programming File
Android 5.0+	/opt/siliconlabs/threadborderrouter/commissioning/thread-commissioning-app-1.01.11.apk

8. Demonstrate the Thread Border Router

8.1 Power On the Border Router

8.2 Connect the Host Computer to the Border Router

1. Connect to the Border Router Wi-Fi access point.
   - SSID: “Silicon Labs Thread XXXX”
   - Passphrase: “solutions”
2. Establish an SSH connection to the Border Router.
   - Username: pi
   - Password: Refer to 5. Install the Border Router Software on the Raspberry Pi.

```
$ ssh pi@192.168.42.1
```

8.3 Reset the Thread Network

1. Send the network reset command to the Border Router.

```
$ echo 'network-management reset' > /dev/border-router-cin
```

2. Confirm border-router-mgmt-app is “up” by tailing its log.

```
$ tail -f /var/log/siliconlabs/border-router-mgmt-app
<22:04:17> Border router is up
```

8.4 Power On Each End Device

Each device has a 64-bit EUI and a fixed, eight-digit, base32 joining passphrase (also called the Joining Device Credential) derived from the EUI, both of which are required by the commissioning application. The joining passphrase and EUI print to stdout of the end device console on boot. Additionally, the EUI is available with the `info` command. These are needed in the next step. See *QSG113: Getting Started with Silicon Labs Thread* for guidance on using the end device console within Simplicity Studio.

8.5 Commission the End Nodes with an Android or iOS Device

1. Using the device on which the commissioning app was installed in Chapter 7, connect to the Thread Border Router Wi-Fi Access Point. The access point will have an SSID of the format “Silicon Labs Thread XXXX,” where "XXXX" is an arbitrary hex number generated randomly after installation of the Border Router Software.
   - SSID = “Silicon Labs Thread XXXX”
   - Passphrase = “solutions.”
2. Launch the commissioning app. The app will search for available Thread networks and request the Thread admin password. The Thread admin password is set at compile time by the Border Router application and printed on stdout immediately after boot. It is “COMMPW1234” for this version of the Thread Border Router.
3. If successful, the commissioning application will indicate that it is ready to accept join credentials by displaying a screen similar to the following:

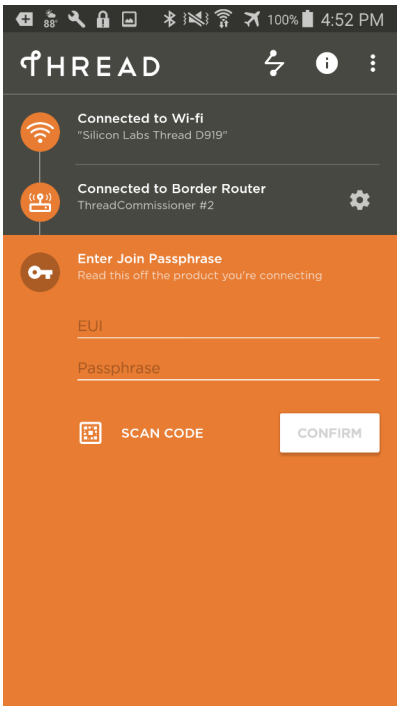

Figure 8.1. Thread Commissioning App

4. To join each end device using the commissioning app:
   a. Enter the EUI and connect code (also called the Passphrase) for a Thread End Node as noted in section 8.4 Power On Each End Device. The commissioning app should pop up a busy screen indicating that it is locating the Thread End Node to join.
   b. Press on the Thread End Node PB1 to issue a net reset command and initiate a scan of all available channels for the Thread Network.
   c. The Thread End Node scans all channels twice and then will go idle unless the Network is found. It is important to enter the connect code on the app first and press Button1/PB1 second. A successful commissioning process is indicated in the commissioning app screen by a large checkmark and a “Device Added” message.

Note: After any network change, such as adding or removing a device or receiving a new IPv6 prefix, the Thread network may take up to two minutes to heal. Devices might not operate correctly during this time.

LED1 on the EFR32MG kits indicates the type of node by using different blink patterns:

1. Very fast blink with pause (either on or off) = joining
2. Fast blink = router
3. Slow blink = end node
4. Alternating fast and slow blink = leader

Hint: To save time typing in the EUI and Passphrase, create a QR code at http://goqr.me using the string format:

```
v=1&cc=EL7TUCB0&eui=000B57FFFE07F6B3
```

Version `v=1`, joining passphrase phrase `cc=EL7TUCB0`, and `EUI=000B57FFFE07F6B3`. Successful generation of this example is shown below:

Figure 8.2. QR Code Example

8.6 Discovering, Controlling, and Monitoring Thread Devices with zclip.js

8.6.1 JavaScript Tools Available for Demonstration and Development

Zclip.js is a Java Script implementation of the ZCL-over-IP (ZCL/IP) application layer defined by the Zigbee Alliance. It is available at https://github.com/SiliconLabs/zclip.js and provides libraries for creating custom ZCL/IP applications.

Silicon Labs Thread Border Router Reference Design ships with two sample zclip.js-based applications to aid in demonstration and development: zclip-cli and zclip-rd server.

- **zclip-cli** is a command line interface that supports ZCL/IP device discovery and control. It is available at https://github.com/SiliconLabs/zclip-cli and may be installed on any computer with IPv6 connectivity to the Border Router.
- **zclip-rd server** is a javascript implementation of a ZCL/IP Resource Directory (RD). An RD provides both a database of ZCL/IP devices currently on the Thread network as well as a server to handle device database registration and queries. It facilitates a number of device management and control tasks including lookup of sleepy devices and remote interaction. zclip-rd should be installed on a border router. Source code for this tool is available at https://github.com/SiliconLabs/zclip-rd

The following sections explain how to use the preinstalled sample applications. See the documentation on github for an explanation of the dependencies between zclip.js, zclip-rd, and zclip-cli.

8.6.2 Device Discovery

Two methods can be used to discover devices on a Thread network: *Resource-Directory Lookup* and *Multicast Discovery*. Both are supported by zclip-cli.

8.6.2.1 Device Discovery Using the RD

Resource Directory Lookup requires that an RD server be present on the border router. Furthermore, devices must register with the RD. Specifically:

1. Devices on the Thread network must have the RD client plugin enabled and must actively register with the RD.
2. Remote RD clients must know the IP address of the border router in order to reach the RD from off-mesh.

Registering with the resource directory

Using Simplicity Studio, connect to the end device and launch a debugging console. Issue the following command:

```
> zcl resource-directory register "<rdIp>" <rdPort>
```

If registration is successful, the following will be returned:

```
> Registering with resource directory
> Registration Success
```

Device lookup using the resource directory

Establish an SSH connection with the border router following the instructions in section 8.2. To discover devices using the RD, issue the following command:

```
pi@br-51B0:~ $ zcl lookup --rdIp <ipv6-address> --rdPort 5689 --page 0 --count 4
```

The paging parameters (page and count) are required when the lookup result exceeds the maximum payload size for a CoAP packet. In practice, the payload handles a count of 4 without issue.

See https://github.com/SiliconLabs/zclip-cli for more usage examples, such as querying for only specific clusters.

8.6.2.2 Device Discovery Using Multicast

When no RD is present, discovery can still be achieved by multi-casting a discovery message across the Thread network. However, not all devices will respond.

1. Devices that have registered with an RD may not respond to multicast discovery messages by design.
2. Sleepy devices are unlikely to be awake to hear multicast discoveries at the time the message is sent.

zclip-cli reverts to multi-cast discovery if no resource directory is specified. For example, devices that support the onOff cluster will respond to the following multicast discovery

```
pi@br-51B0:~ $ zcl lookup --cluster onOff
```

8.6.3 Device Interaction

Once a device has been discovered, it can be read or controlled using zclip-cli. The target device can be specified by its IPv6 address as follows:

```
zcl cmd <cluster> <command> --ip <ip> [arguments]
```

For example:

```
zcl cmd levelControl move --ip ::1 --moveMode 1 --rate 255
```

One disadvantage of using the ipv6 address in this way is that the address can change at any time. A preferred way to specify the target device is using its UID as follows:

```
zcl cmd <cluster> <command> --uid <uid> --rdIp <RDIP> --rdPort <RDPort> [arguments]
```

For example:

```
zcl cmd levelControl move --uid P7BU0eh27b2f5f0IC0GLL7uHum_DtmRVe73umRQCAeY --rdIp ::1 --rdPort 5689 --
moveMode 1 --rate 255
```

The UID-specified method requires that we provide the location of a RD. This is because UIDs are not usable for global routing. Therefore, when you attempt to identify the target device using UID, zclip-cli actually performs two transactions behind the scenes:

1. zclip-cli queries the RD in order to find the IPv6 address that matches the UID.
2. zclip-cli then sends the message to the target using the IPv6 address.

The purpose of first querying the RD is that it should have the correct IPv6 address of the target, because all devices are supposed to update their registration with the RD any time their IPv6 address changes.

See https://github.com/SiliconLabs/zclip-cli for more usage examples, including attribute reads and binding.

9. Next Steps

The next steps include modifying and compiling the Thread End Node firmware and Thread Border Router software. Refer to *UG116: Developing Custom Border Router Applications* for more information.

10. Resources

10.1 Getting started with Thread

http://www.silabs.com/products/wireless/Pages/thread-getting-started.aspx

10.2 Thread Training Center

http://www.silabs.com/products/wireless/Pages/thread-networking-learning-center.aspx

10.3 Documentation

- *UG110: Ember® EM35x Development Kit User Guide*
- *QSG113: Getting Started with Silicon Labs Thread*
- *UG103-11: Application Development Fundamentals: Thread*
- *UG116: Developing Custom Border Router Applications*

10.4 Community & Support

http://community.silabs.com/

http://www.silabs.com/support