UG116: Developing Custom Border Router Applications

A Thread border router is a device in a Thread network that provides connectivity to adjacent networks on other physical layers. The border router provides services for devices within the Thread network, including routing services for off-network operations. Silicon Labs provides a Border Router Add-On Kit containing a Raspberry Pi device, example applications, and the underlying APIs required to build border router software. This user guide provides information for developing custom border router applications.

KEY FEATURES

- Architecture
- Operation
- Build and installation instructions
- Border Router configuration
- Test router configuration
- Utilities, files and services
- Resources

1 Introduction

This user guide is intended for software engineers who wish to develop a Thread Border Router System, and who are familiar with the Thread specification, Silicon Labs Thread stack and Silicon Labs development kits. The hardware referenced in this document is available with the Thread Border Router Add-On Kit, part number RD-0004-0201.

This user guide assumes familiarity with these documents:

- QSG102, *Thread Border Router Add-On Kit Quick Start Guide*
- QSG113, *Getting Started with Silicon Labs Thread*
- UG103.11, *Application Development Fundamentals: Thread*.
- AN1010, *Building a Customized NCP Application*

This user guide addresses the following topics:

- **Architecture**

  Shows the components, data paths, and management paths between the Thread commissioner, Thread Border Router host and NCP, and Thread end nodes.

- **Operation**

  Describes the operation of the Thread Border Router System, including commissioning, discovery and end-to-end IPv6 communication.

- **Build and Installation Instructions**

  Describes the build and installation procedure for the IP modem, IP driver, commissioning, border router and web server applications.

- **Border Router configuration**

  Provides a reference to explain the modifications the silabs-border-router package installation makes to the Raspbian Jessie Lite operating system installation.

- **Test router configuration**

  Provides a procedure to install the silabs-test-router package on the Border Router.

- **Utilities, Files and Services**

  Contains a listing that may be useful for Border Router configuration and development

- **Resources**

  Includes additional information available for software engineers.

2 Architecture

This section describes the components and features of the Thread Border Router System, including the Thread commissioner, Thread Border Router host and NCP, and Thread end nodes.

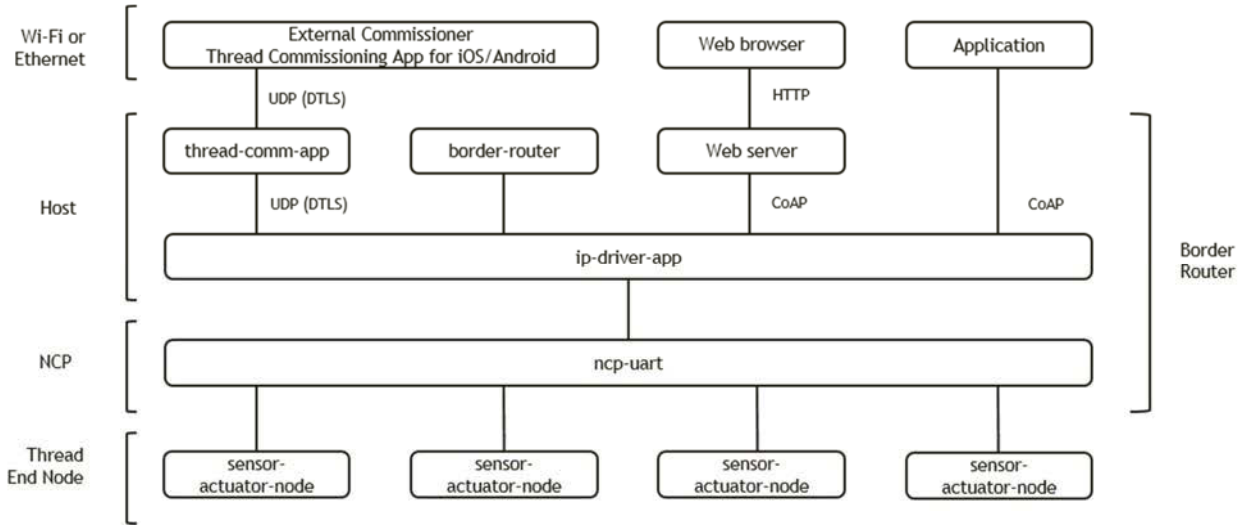


Figure 1. Border Router System Architecture

2.1 Thread Commissioning App for iOS/Android

The Thread commissioning app for iOS/Android is a utility and reference design for commissioning Thread devices. It exchanges commissioner data via Wi-Fi with the `thread-comm-app` on the Thread Border Router host and features a tool to securely commission Thread end nodes on the 802.15.4 Thread network.

The external Thread commissioner app and source for iOS/Android is provided by the Thread Group and can be acquired as described in QSG102, *Thread Border Router Add-On Kit Quick Start Guide*. The steps for building the application are outside the scope of this document.

2.2 Border Router Commissioning Application (`thread-comm-app`)

The Border Router commissioning application (`thread-comm-app`) on the Border Router host manages the commissioning process. Its features include:

- Initiating mDNS advertisement with the Avahi service to the external Thread commissioning app for iOS/Android.
- Acting as commissioner when there is no external commissioner.
- Acting as the security and authentication module on the host.
- Running the DTLS/JPAKE handshake required for Thread end node commissioning.

2.3 Border Router IP driver Application (`ip-driver-app`)

The IP driver application (`ip-driver-app`) on the Border Router acts as the main dispatch for the IP modem. Its features include the ability to act as an intermediary, similar to a multiplexer / de-multiplexer, that can read from and write to different streams over a single serial connection between the host and NCP.

2.4 Border Router IP Modem Application (`ncp-uart`)

The IP modem application (`ncp-uart`) on the Border Router runs the Thread stack software. Its features include a serial host interface that can run on Silicon Labs devices such as EM35x or EFR32MG. In the case of the Thread Border Router Add-On kit, the NCP is the CEL EM3588 USB Thread Adapter.

2.5 Thread End Node Sensor/Actuator Application (`sensor-actuator-node`)

The sensor/actuator application (`sensor-actuator-node`) on the Thread end nodes provides I/O for sensors, buttons and indicators. It exchanges data and management packets via the Thread network with the `ncp-uart` application on the Border Router NCP. Its features include:

- Secure commissioning by the Thread commissioning app for iOS/Android.
- LED and buzzer actuator control.
- Push button and temperature telemetry reporting.

2.6 Border Router Application (`border-router`)

The Border Router application (`border-router`) is a system management application. It features include:

- Thread network configuration.
- Device discovery.
- Removal of Thread end nodes from the Thread network.
- Removal of the Thread Border Router from the Thread network.

2.7 Web Server

The web server application on the Border Router host serves a user interface for the border-router application and is included to help with demonstration and diagnostics. It is similar in concept to a router administration server. Its features include:

- Exposing border-router application features.
- Controlling and monitoring Thread end nodes.
- Removing devices from view.
- Displaying system IP addresses.
- Acting as the Node.js back-end server.
- Providing React UI front-end implementation for rendering on computer or mobile device.

2.8 Web Browser

The web browser renders the web server user interface. Demonstration with the web browser is described in QSG102, *Thread Border Router Add-On Kit Quick Start Guide.*

2.9 Application

The application refers to an iOS/Android mobile device app, web browser plug-in such as Copper for the Mozilla browser, or any other application capable of communicating with the `ip-driver-app` via CoAP. Demonstration with Copper plugin is described in QSG102, *Thread Border Router Add-On Kit Quick Start Guide.*

3 Operation

This section describes the operation of the Thread Border Router System, including commissioning, discovery and end-to-end IPv6 communication.

3.1 Commissioning

1. The `border-router` application establishes the commissioner key and requests the `ncp-uart` application form a Thread network.
2. The external commissioner establishes a connection with the `thread-comm-app` using the commissioner key.
3. The external commissioner performs the DTLS/JPAKE handshake to commission the Thread end node.

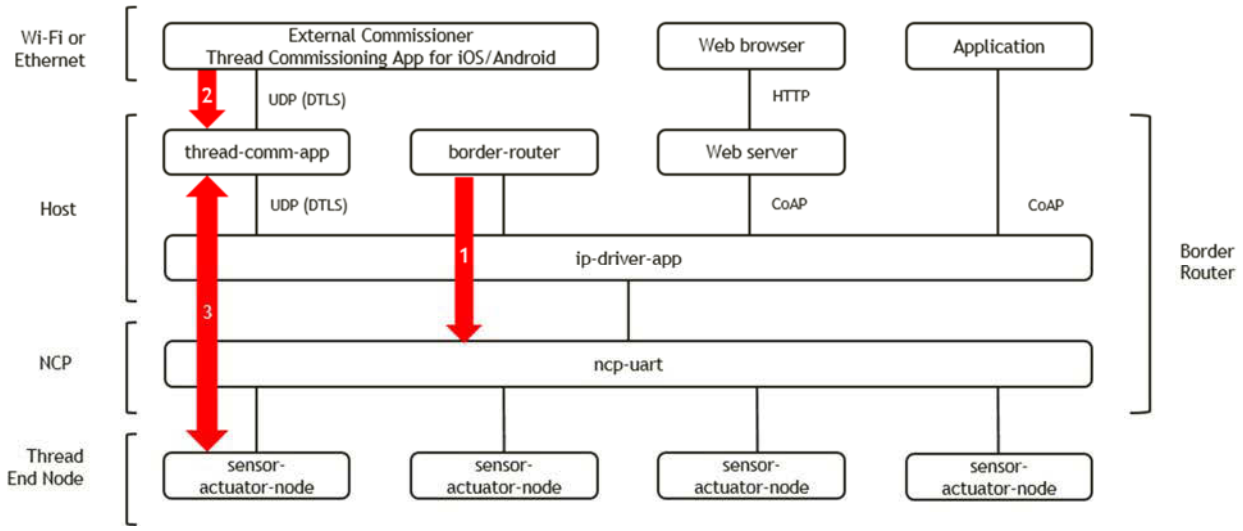


Figure 2. Commissioning

3.2 Discovery

1. An application sends a discovery request via CoAP to the `border-router` application.
2. The `border-router` application sends a discovery multicast request to the `ncp-uart` application.
3. Each Thread end node responds to the application with its global IPv6 address and device type.

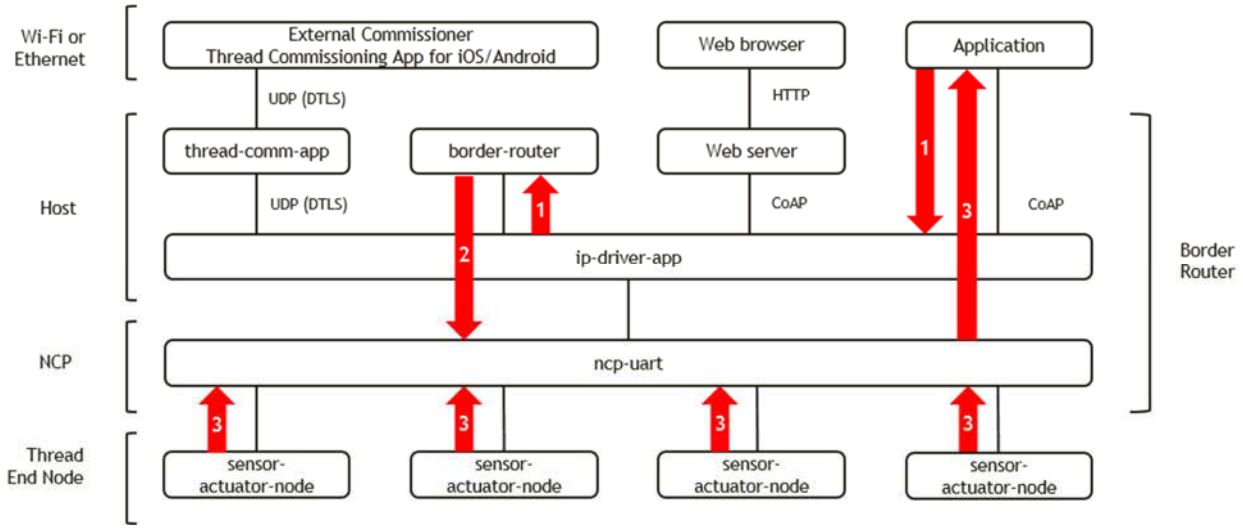


Figure 3. Discovery

3.3 End-to-End IPv6 Communication

1. The web server, or an application such as the Copper plugin for Mozilla, sends and receives messages via CoAP to and from the IPv6 address of the desired Thread end node. This assumes commissioning and discovery have completed.

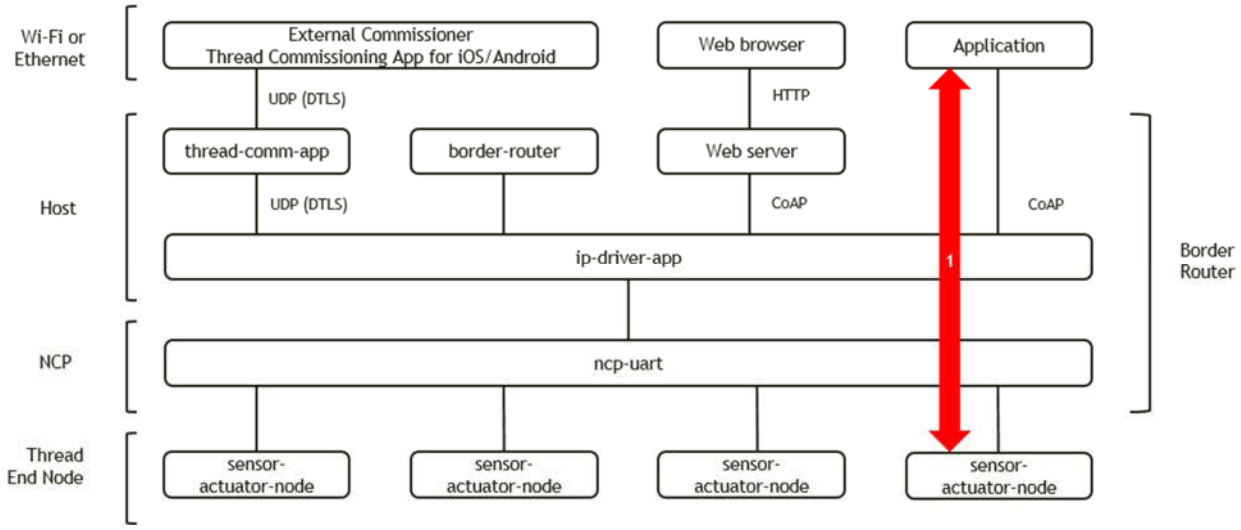


Figure 4. End-to-End IPv6 Communication

4 Build and Installation Instructions

This section describes the build and installation procedure for the IP modem, IP driver, commissioning, border router, and web server applications. The following steps assume that you have installed Simplicity Studio, the Silicon Labs Thread stack, and the IAR-EWARM compiler. Refer to QSG113, *Getting Started with Silicon Labs Thread*, for a detailed tutorial.

4.1 Build the Border Router IP Modem Application

The Silicon Labs Thread stack supports building NCP applications in Simplicity Studio AppBuilder, with customizations for target hardware, initialization, main loop processing, event definition and handling, and host/NCP command extensions. Refer to AN1010, *Building a Customized NCP,* for a detailed set of build instructions. Refer to CEL 0011-09-07-00-000 (Issue G) for a detailed set of programming instructions for using an ISA3 debug adapter to install the application and bootloader. **Note:** The bootloader for the CEL EM3588 USB Thread Adapter is located at tool/bootloader-em3588/serial-uart-bootloader, relative to root of the Silicon Labs Thread stack installation.

4.2 Build the Border Router IP Driver Application

1. Copy the Thread stack to the Border Router.
2. It is not necessary to generate a project with AppBuilder because there are no configuration options required.
3. Make the application from the root of the Thread stack directory with this command:

   ```
   $ make -f app/ip-ncp/ip-driver-app.mak
   ```

4. Copy the application to the executable directory from the root of the Thread stack directory with this command:

   ```
   $ cp build/ip-driver-app/ip-driver-app /opt/siliconlabs/threadborderrouter/bin
   ```

4.3 Build the Border Router Commissioning Application

1. Obtain the source code for the Rijndael cipher.
2. The `thread-comm-app` requires the AES-128 cipher. Silicon Labs cannot provide this software implementation of AES-128 because it is classified as strong cryptography and subject to export restrictions. An implementation of Rijndael known to be compatible with the Thread commissioning application is:
   - http://www.efgh.com/software/rijndael.zip (`md5: cd49617fa6593d2ab67a68f64ede2d78`)
3. The cipher consists of the following files:
   - `rijndael-alg-fst.c`
   - `rijndael-alg-fst.h`
   - `rijndael-api-fst.c`
   - `rijndael-api-fst.h`
4. The header files are included in the Silicon Labs Thread installation to assist you in locating the corresponding source files. Once Rijndael has been acquired, copy the source files to the following locations relative to the root of the Thread stack installation to complete the configuration:
   - `hal/micro/generic/aes/rijndael-alg-fst.c`
   - `hal/micro/generic/aes/rijndael-api-fst.c`
5. Copy the Thread stack to the Border Router.
6. It is not necessary to generate a project with AppBuilder because there are no configuration options required.
7. Make the application from the root of the Thread stack directory with this command:

   ```
   $ make -f app/thread-commissioning/thread-comm-app.mak
   ```

8. Copy the application to the executable directory from the root of the Thread stack directory with this command:

   ```
   $ cp app/thread-commissioning/thread-comm-app /opt/siliconlabs/threadborderrouter/bin
   ```

4.4 Build the Border Router Application

1. Create and generate the border-router example.
2. Select generation directory “app/thread/sample-app/border-router/border-router” relative to the stack and select “Unix host” as the target architecture. This will generate the project in place within the Thread stack.
3. Copy the Thread stack to the Border Router.
4. Make the application from the root of the Thread stack directory with this command:

```
$ make -C app/thread/sample-app/border-router/border-router
```

5. Copy the application to the executable directory from the root of the Thread stack directory with this command:

```
$ cp app/thread/sample-app/border-router/border-router/build/exe/border-router
/opt/siliconlabs/threadborderrouter/bin
```

4.5 Build the Web Server Application

The website React user interface source code is located at /opt/siliconlabs/threadborderrouter/src/reactui and the node.js server source is located at /opt/siliconlabs/threadborderrouter/webserver. Follow the instruction below to build these components.

1. Install dependencies:

```
$ cd /opt/siliconlabs/threadborderrouter/src/reactui
$ sudo npm install -g gulp
$ sudo npm install
```

2. Generate the React user interface, copy to the server directory and restart the Apache server with these commands:

```
$ sudo gulp build-dev
$ sudo service apache2 stop
$ sudo rm -r /var/www/html/*
$ sudo cp -r /opt/siliconlabs/threadborderrouter/src/reactui/dist/* /var/www/html
$ sudo service apache2 start
```

3. Restart the node.js server with the command:

```
$ sudo service border-router-apps {start | stop | restart}
```

5 Silicon Labs Thread Border Router Configuration

This section provides a reference to the modifications the silabs-border-router package makes to the Raspbian Jessie Lite operating system. The steps for installing the silabs-border-router package are described in QSG102, *Thread Border Router Add-On Kit Quick Start Guide*. The subnet assignments for the silabs-border-router package are:

Interface	Type	IPv6 Subnet	Type	IPv4 Subnet
Wi-Fi (wlan0)	SLAAC	2001:0db8:8569:b2b1::/64	DHCP	192.168.42.1/24
Ethernet (eth0)	DHCP client	Dynamically assigned	DHCP client	Dynamically assigned
802.15.4 (Thread)	SLAAC	2001:0db8:0385:9318::/64		Not applicable

The wlan0 and Thread subnets are fixed by the Border Router configuration, however, the Border Router configuration could be modified to subdivide a subnet assignment received by the eth0 DHCP server.

IMPORTANT: The 2001:0db8::/32 IPv6 subnets are classified as “documentation only” and should not be used on the IPv6 internet.

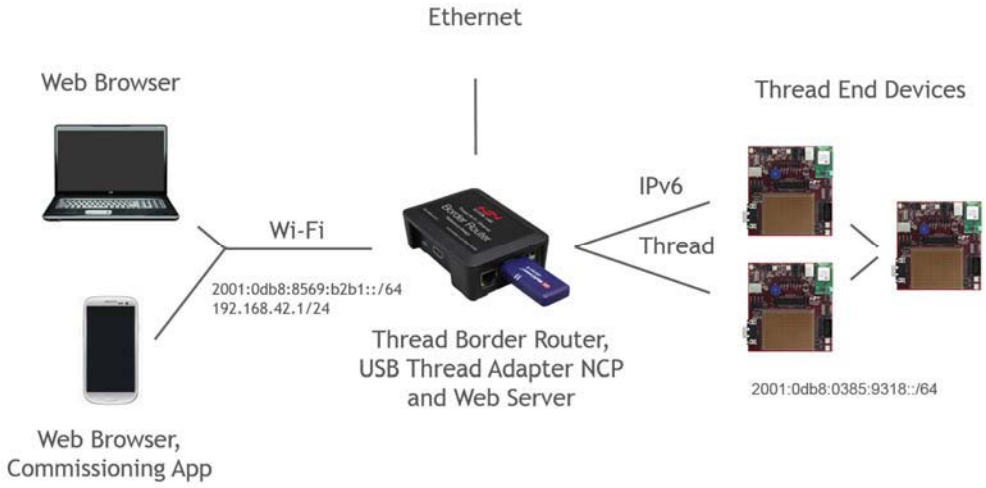


Figure 5. Thread Border Router Configuration

5.1 Load IPv6 and CP210x drivers

The IPv6 and cp210x kernel modules must be loaded for the Linux networking stack to process IPv6 traffic.

1. Add the following to /etc/modules to enable IPv6 on boot:

```
ipv6
cp210x
```

5.2 Configure Network Interfaces

All interfaces on which addresses are distributed must have an IP address within the subnet they distribute so that clients may communicate with the router. The Border Router is configured as a router on wlan0 and therefore requires a static IP address on this interface.

1. Replace /etc/network/interfaces with the following:

```
auto lo wlan0

iface lo inet loopback
iface lo inet6 loopback
iface eth0 inet dhcp

allow-hotplug wlan0 eth0

iface wlan0 inet static
```

```
  address 192.168.42.1
  netmask 255.255.255.0

iface wlan0 inet6 manual
  post-up ip -6 addr add  2001:0db8:8569:b2b1::1/64 dev wlan0
  post-down ip -6 addr del 2001:0db8:8569:b2b1::1 dev wlan0

up iptables-restore < /etc/iptables.ipv4.nat
```

2. Reboot, or manage the service with this command:

```
$ sudo service networking restart
```

5.3 Configure IPv6 Packet Forwarding and Router Advertisements

The Border Router must forward IPv6 packets between interfaces so that IPv6 devices on different interfaces such as wlan0 and eth0 may communicate with each other. The Border Router must also reject router advertisements for interfaces on which it is a router, such as wlan0, and accept router advertisements for interfaces on which it is not a router, such as eht0.

1. Add the following to /etc/sysctl.d/30-silabs.conf to configure settings automatically at boot:

```
net.ipv6.conf.default.forwarding=1
net.ipv6.conf.all.forwarding=1
net.ipv6.conf.wlan0.accept_ra=0
net.ipv6.conf.eth0.accept_ra=2
```

   The first commands enables forwarding as a default behavior on all interfaces. The second command causes the Border Router to search for next-hop destinations for incoming packets not addressed to the Border Router's physical address. If a next-hop is found in the Linux routing tables, the packet will be forwarded by the networking stack to the corresponding interface. The third command prohibits accepting router advertisements on wlan0, the interface on which the Border Router acts as a routing device. The last command accepts router advertisements on eth0, the interface intended to receive an ipv6 subnet allocation from an external router or ISP. These settings allow the Border Router to act as a router on wlan0 while still obtaining a network prefix from eth0.

   An "accept_ra" value of '0' disables the acceptance of router advertisements on that interface, and is the default behavior for all interfaces on a router unless overridden. The value '1' enables acceptance on an interface, provided forwarding is not enabled. The value '2' notifies the Linux kernel to override its default behavior of disabling the acceptance of router advertisements when it is operating as a router (forwarding packets).

5.4 Configure IPv4 Packet Forwarding

The Border Router must forward IPv4 packets between interfaces so that IPv4 devices on different interfaces such as wlan0 and eth0 may communicate with each other.

1. Modify /etc/sysctl.d/30-silabs.conf to include the following:

```
net.ipv4.ip_forward=1
```

2. Update the IPv4 ip tables with these commands (effective until the next boot):

```
$ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
$ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

3. Configure IPv4 masquerading on boot with these commands:

```
$ sudo iptables-save > /etc/iptables.ipv4.nat
$ sudo echo "up iptables-restore < /etc/iptables.ipv4.nat" >> /etc/network/interfaces
```

   The first command causes `iptables` to save its current configuration in a file located at: /etc/iptables.ipv4.nat. The second command appends /etc/network/interfaces with a command that causes `iptables` to restore this saved configuration after networking starts on your system.

5.5 Configure Domain Name Masquerader (dnsmasq)

On an IPv6 network, devices acting as routers advertise the mechanism used to select a unique IPv6 address, as well as a subnet prefix for their network. There are two primary methods for automatic address assignment: DHCPv6 (Dynamic Host Configuration Protocol Version 6) and SLAAC (Stateless Address AutoConfiguration). DHCPv6 operates similarly to DHCP for IPv4 networks, where a central device maintains an address space and leases addresses to end nodes for a certain amount of time based upon a request-response handshake. SLAAC is a new protocol designed to eliminate the need for DHCP servers. With SLAAC, devices on an IPv6 network can query their neighbors to determine if a potential address is being used and, if the address is unused, the device can notify its neighbors that it will be available immediately at that address. Because IPv6 anticipates a wide variation in network configuration, where IPv4, IPv6, DHCPv6 and SLAAC may all be operating simultaneously, router advertisement packets serve as a notification of a network's capabilities to joining nodes. To enable router advertisements on a device, utilities such as `radvd` (the Router Advertisement Daemon) or `dnsmasq` (the Domain Name Masquerader) can be used. The Border Router uses `dnsmasq` as described below.

The domain name masquerader `dnsmasq` provides an alternative router advertisement daemon to `radvd`, and provides SLAAC as well as DHCP (Dynamic Host Configuration Protocol) to assign IPV6 addresses. In comparison to SLAAC, DHCPv6 can provide a specific range of IP addresses to end nodes, which they obtain through negotiating with the Border Router.

1. Modify /etc/dnsmasq.d/silabsap.conf with these commands:

```
interface=wlan0
dhcp-range=2001:DB8:8569:B2B1::,ra-only,10m
dhcp-range=192.168.42.50,192.168.42.150,12h
dhcp-authoritative
enable-ra
```

   The configuration above enables router advertisements (allowing SLAAC), and then provides a prefix in the 2001:DB8:8569:B2B1:: range. The directive `dhcp-authoritative` configures `dnsmasq` to negatively-acknowledge any devices requesting DHCP addresses that are not from the range it is assigning. This causes all devices on the Border Router network to eventually obtain addresses from the ranges specified by `dnsmasq` rather than other DHCP servers operating on the link. The directive `ra-only` specifies that `dnsmasq` should only provide SLAAC addresses, but additional settings can be used to provide DHCPv6 if-desired. If Domain Name Resolution, or DHCPv6 are required for your solution, consult the documentation for `dnsmasq` at http://www.thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html

2. Manage `dnsmasq` with the following command:

```
$ sudo service border-router-apps {start | stop | restart }
```

5.6 Configure Neighbor Discovery Protocol Proxy Daemon (ndppd)

Neighbor Discovery Protocol is specified in RFC 4861 and is a foundational component of IPv6 reachability, routing, and SLAAC. In order to discover neighbors on an IPv6 network, a router issues 'Neighbor Solicitations' designed to discover if a node exists at a target IPv6 address. Neighbor Solicitations used in this fashion are similar to the use of 'ARP' in IPv4: a node is expected to reply to any Neighbor Solicitations for its address with a 'Neighbor Advertisement.' Because Thread 6LowPAN only implements a subset of IPv6 in order to save power, mesh devices do not provide Neighbor Discovery Protocol. Instead, the Border Router acts as a proxy for all Neighbor Advertisements directed to its mesh devices.

1. Modify /etc/ndppd.conf as follows:

```
route-ttl 30000
proxy eth0 {
  router yes
  timeout 500
  ttl 3000
  rule 2001:0db8:0385:9318::/64 {
    static
  }
}
```

2. Reboot, or manage npdpd with the following command:

```
$ sudo service border-router-apps {start | stop | restart}
```

5.7 Avahi

Avahi is an implementation of the Apple mDNS bonjour protocol, which allows devices to register and advertise a human-readable name with other devices on the local network. The Border Router publishes the border router name with the `avahi` daemon via IPv4 so that the iOS or Android commissioning is aware of the Border Router. The configuration file is /etc/avahi/avahi-daemon.conf and does not require modification. The `avahi` daemon is managed by the border-router-app service.

6 Test Router Configuration

Because IPv6 is not fully deployed, a Border Router can be converted into a Test Router, which will allow operation of an IPv6 network independent of the Internet. This section provides a procedure to install the `silabs-test-router` package. The subnet assignments for the test router are:

Interface	IPv6 Type	IPv6 Subnet	IPv4 Type	IPv4 Subnet
Wi-Fi (wlan0)	SLAAC	2001:0db8:8569:b2b4::/64	DHCP	192.168.141.1/24
Ethernet (eth0)	SLAAC	2001:0db8:8569:b2b3::/64	DHCP	192.168.222.1/24

IMPORTANT: The 2001:0db8::/32 IPv6 subnets are classified as “documentation only” and should not be used on the IPv6 internet.

IMPORTANT: The test router Ethernet (eth0) should not be connected to a network that is not expecting IPv6 or IPv4 address assignment, such as a corporate network. To safely connect to a corporate network, stop the `dnsmasq` service with the command:

```
$ sudo service dnsmasq {start | stop | restart}
```

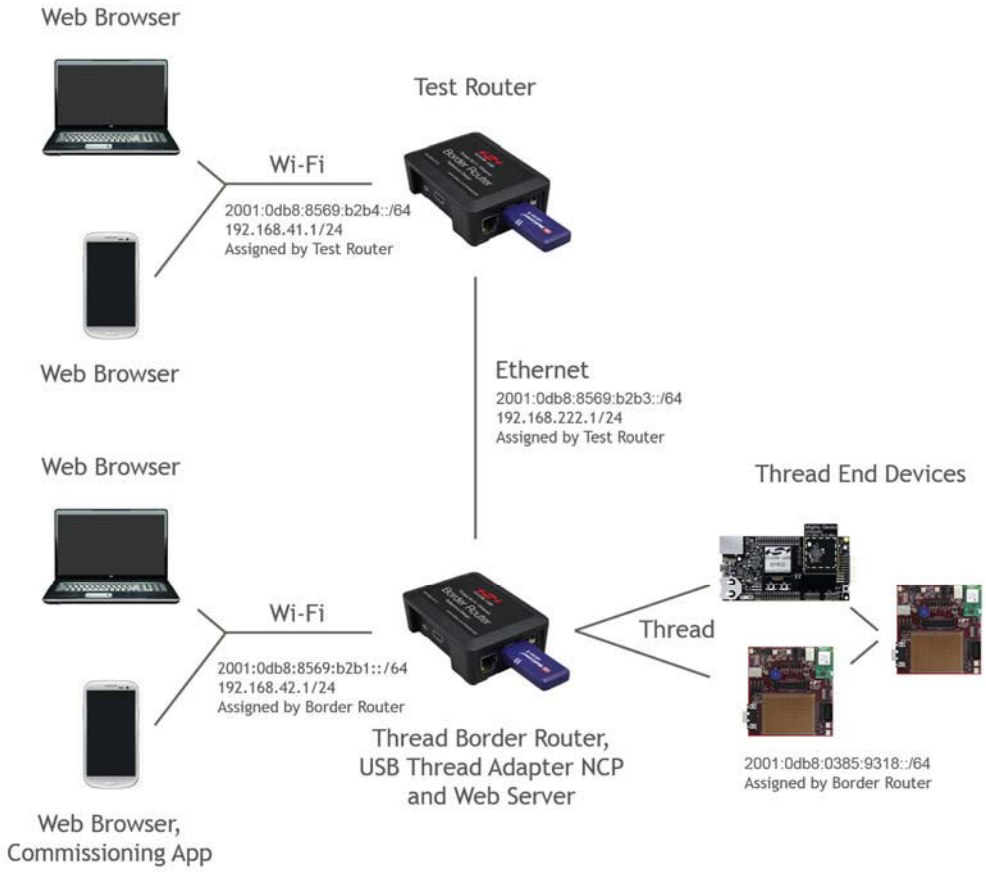


Figure 6. Test router configuration

6.1 Install the Test Router Operating System and Packages

1. Power down the Thread Border Router and remove the SD card.
2. Install the Raspbian Jessie Lite operating system on the SD card as described here: https://www.raspberrypi.org/downloads/raspbian/. **Note:** All Border Routers shipped after August 1, 2016 include an SD card with the Raspbian Jessie Lite operating system pre-installed.
3. Install the SD card in the Thread Border Router and power it on.
4. Connect the Raspberry Pi’s Ethernet port to the Internet with an Ethernet cable.

5. Login with the default username “pi” and password “raspberry.”
6. Append the following text to the end of /etc/apt/sources.list:

```
deb http://devtools.silabs.com/solutions/apt/ jessie main
```

7. Configure the keys.

```
$ sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys DE864524
```

8. Install the silabs-border-router package. Note that you will be asked to accept the Thread Border Router license agreement.

```
$ sudo apt-get update
$ sudo apt-get install silabs-test-router
$ sudo reboot
```

7 Utilities, Files and Services

The following utilities, files and services may be useful for Border Router configuration and development.

7.1 Write/Read a Disk Image to/from the microSD Card

It is possible to write a disk image to the microSD card or read a disk image from the microSD card using a Windows, Mac or Linux computer. The reason to do this would be to copy the operating system disk image to the microSD card for first time installation, or to read the file system from microSD card to archive or duplicate the disk image. From a PC, use Win32DiskImager and from Linux/Mac use the `dd` command.

7.2 Expand the microSD Card File System

The file system may be expanded to the full capacity of the microSD card. The reason to do this would be to enable the full microSD card to be used by the Border Router. From the Border Router use the `sudo raspi-config` command.

7.3 Change the Keyboard Settings

The Raspberry Pi uses an English (UK) keyboard mapping by default. Users may need to change this setting in order to access special characters. From the Border Router use the `sudo raspi-config` command.

7.4 Remote Login to the Border Router

It is possible to login remotely to the Border Router once connected to the Border Router Wi-Fi access point or Ethernet. The reason to do this would be to avoid needing a separate monitor and keyboard. From a PC, open an SSH session using PuTTY at 192.168.42.1, port 22. From a Linux/Mac, use the `ssh` and `ssh-copy-id` commands to open a session to pi@192.168.42.1. Login with the default username “pi” and password “raspberry.”

7.5 Transfer Files To and From the Border Router

It is possible to transfer files between a Windows, Mac or Linux computer and the Border Router once connected to the Border Router Wi-Fi access point or Ethernet. The reason to do this would be to copy the Thread stack to the Border Router for the purpose of building applications described in section Build and Installation Instructions. From a PC, open a session using WinSCP at 192.168.42.1, port 22. From a Linux/Mac, use the `scp` command to open a session to pi@192.168.42.1. Login with the default username “pi” and password “raspberry.”

7.6 Controlling the `thread-comm-app`, `ip-driver-app` and `border-router` Applications

The `silabs-scripts` shell script exposes a useful capability of running applications in the foreground or background. The applications are normally run in the background, however, running the border-router application may be useful for command line control. Stop and then start the border-router in the foreground with these commands:

```
$ /opt/siliconlabs/threadborderrouter/bin/silabs-scripts -b stop
$ /opt/siliconlabs/threadborderrouter/bin/silabs-scripts -b fg
```

7.7 Files

File	Purpose
/var/www/html/assets/version.json	Specifies major, minor and sub revision of the silabs-border-router package.
/opt/siliconlabs/threadborderrouter/ip-driver-app	IP driver application
/opt/siliconlabs/threadborderrouter/thread-comm-app	Thread commissioning application
/opt/siliconlabs/threadborderrouter/border-router	Border router application
/var/log/siliconlabs/ip-driver-app	IP driver application log file (not size managed)
/var/log/siliconlabs/thread-comm-app	Thread commissioning application log file (not size managed)
/var/log/siliconlabs/border-router	Border router application log file (not size managed)
/etc/ndppd.conf	ndppd configuration file.
/etc/dnsmasq.d/silabsap.conf	Dnsmasq configuration file.
/etc/sysctl.d/30-silabs.conf	Sysctl configuration file.
/etc/hostapd/hostapd.conf	Host APD configuration file.
/etc/siliconlabs/border-router.conf	Border router configuration file.

7.8 Services

The following services may be useful for Border Router configuration and development.

File	Purpose
border-router-apps	The ip-driver-app, thread-comm-app, and border-router applications are managed by this service.
dnsmasq	Dnsmsaq service.
networking	Networking service.
hostapd	Host Wi-Fi access point service.

8 Resources

8.1 Getting started with Thread

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

8.2 Thread Training Center

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

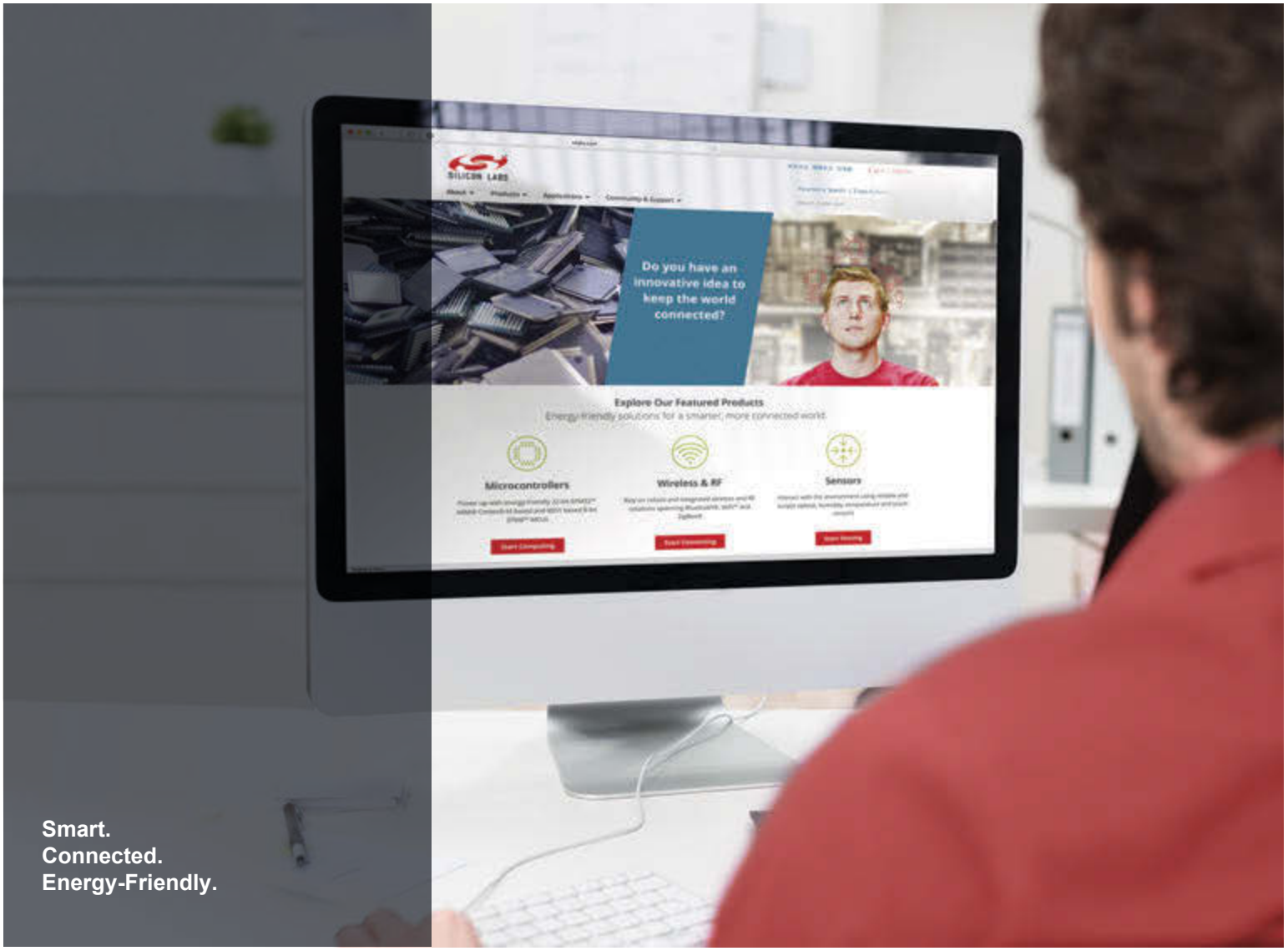
8.3 Documentation

- QSG101, *EM35x Development Kit Quick-Start Guide* (included in development kit)
- QSG102, *Thread Border Router Add-On Kit Quick Start Guide*
- QSG113, *Getting Started with Silicon Labs Thread*
- UG103.11, *Application Development Fundamentals: Thread*
- AN1010, Building a Customized NCP Application
- CEL 0011-09-07-00-000 (Issue G)

8.4 Community & Support

http://community.silabs.com/

http://www.silabs.com/support



Products
www.silabs.com/products

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS

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

http://www.silabs.com