UG495: Silicon Labs Wi-SUN Developer’s Guide

This document is a reference for those developing applications using the Silicon Labs Wi-SUN (Wireless Smart Ubiquitous Network, Field Area Network) SDK (Software Development Kit). The guide covers the (Wi-SUN) stack architecture, application development flow, steps to configure the application Wi-SUN radio settings and advanced debug features. This version applies to the Silicon Labs Wi-SUN SDK version 1.x.x and higher.

The purpose of this document is to fill in the gaps between the Silicon Labs Wi-SUN Field Area Network (FAN) API reference, Gecko Platform references, and documentation for the target EFR32xG part. This document provides details that will help developers optimize their application for their target environment.

KEY POINTS

• Wi-SUN SDK stack, including firmware and application project structure, and software components
• Application development guidelines
• Wi-SUN radio configuration
• Advanced tools to test and debug a Wi-SUN application
# Table of Contents

1 Introduction................................................................................................................................................................. 1

2 Wi-SUN FAN Stack.............................................................................................................................................................. 2
   2.1 Firmware Structure....................................................................................................................................................... 2
   2.2 Application Project Structure............................................................................................................................................ 3
      2.2.1 Wi-SUN Files Library Files .................................................................................................................................. 3
      2.2.2 RAIL.......................................................................................................................................................................... 3
      2.2.3 EMLIB and EMDRV.................................................................................................................................................. 3
      2.2.4 Mbed TLS................................................................................................................................................................ 3
   2.3 Optional Software Components .................................................................................................................................. 4

3 Wi-SUN Application Development .................................................................................................................................... 5
   3.1 Responding to Wi-SUN Events........................................................................................................................................ 5
   3.2 Implementing Application Logic......................................................................................................................................... 5
   3.3 Changing Operating System.............................................................................................................................................. 5
   3.4 Using a Different Development Environment................................................................................................................. 6
   3.5 Wi-SUN Stack Heap Requirement................................................................................................................................... 7

4 Wi-SUN Configurator ............................................................................................................................................................ 8
   4.1 Application Panel .............................................................................................................................................................. 8
   4.2 Security Panel.................................................................................................................................................................... 9
   4.3 Radio Panel....................................................................................................................................................................... 9
   4.4 Changing the Default Wi-SUN Radio Configuration ...................................................................................................... 10

5 Testing and Debugging.......................................................................................................................................................... 11
   5.1 Access Debug Traces from the Wi-SUN Stack .................................................................................................................. 11
   5.2 Export Wi-SUN Traces to Wireshark............................................................................................................................... 11
   5.3 Connect the Wi-SUN Network to Another IP Network ................................................................................................ 13
1 Introduction

This document contains information for anyone developing applications in the Silicon Labs Wi-SUN SDK. It assumes that the current version of the Silicon Labs Wi-SUN SDK has been installed and that the developer is familiar with creating and flashing applications, and with the functionality available as a starting point in the example files contained in the SDK. If you are not familiar with these items, and are just getting started, see the Simplicity Studio 5 User’s Guide and QSG181: Silicon Labs Wi-SUN Quick-Start Guide. For more information about configuring a Wi-SUN network, see AN1332: Silicon Labs Wi-SUN Network Setup and Configuration.

The Silicon Labs Wi-SUN API reference that matches the installed SDK is available on the Simplicity Studio DOCUMENTS tab. All versions are available at https://docs.silabs.com/wisun/latest/wisun-stack-api/.
2 Wi-SUN FAN Stack

2.1 Firmware Structure

The following figure describes the high-level firmware structure. The developer creates an application on top of the stack, which Silicon Labs provides as a precompiled object-file, enabling the Wi-SUN connectivity for the end-device.

![Figure 2.1. Wi-SUN Stack Architecture Block Diagram](image)

The Wi-SUN stack contains following blocks:

- Wi-SUN stack – Wi-SUN functionality consisting of an IP stack, MAC layer, the routing protocol (RPL), and security manager.
- Wi-SUN RF test plugin – Optional software component to add an API to perform RF tests (for example, create an RF tone).
- Wi-SUN Util Functions – Optional software component to add helper functions to inform the application about the Wi-SUN PHY configured in the RAIL configuration file.
2.2 Application Project Structure

This section explains the application project structure and the mandatory and optional resources that must be included in the project.

2.2.1 Wi-SUN Files Library Files

The Wi-SUN stack libraries are summarized in the following table.

<table>
<thead>
<tr>
<th>Wi-SUN stack library name</th>
<th>To use with</th>
</tr>
</thead>
<tbody>
<tr>
<td>libwisun_router_efr32xg1x_micrium_gcc_debug.a</td>
<td>Wi-SUN router with Micrium OS, GCC, with debug traces</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_micrium_gcc_release.a</td>
<td>Wi-SUN router with Micrium OS, GCC, no debug trace</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_micrium_iar_debug.a</td>
<td>Wi-SUN router with Micrium OS, IAR, with debug traces</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_micrium_iar_release.a</td>
<td>Wi-SUN router with Micrium OS, IAR, no debug trace</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_freertos_gcc_debug.a</td>
<td>Wi-SUN router with FreeRTOS, GCC, with debug traces</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_freertos_gcc_release.a</td>
<td>Wi-SUN router with FreeRTOS, GCC, no debug trace</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_freertos_iar_debug.a</td>
<td>Wi-SUN router with FreeRTOS, IAR, with debug traces</td>
</tr>
<tr>
<td>libwisun_router_efr32xg1x_freertos_iar_release.a</td>
<td>Wi-SUN router with FreeRTOS, IAR, no debug trace</td>
</tr>
<tr>
<td>libwisun_rcp_efr32xg1x.a</td>
<td>Radio coprocessor (Linux border router)</td>
</tr>
<tr>
<td>libwisun_mac_efr32xg1x.a</td>
<td>MAC layer for the Radio coprocessor (Linux border router)</td>
</tr>
</tbody>
</table>

2.2.2 RAIL

The Wi-SUN stack uses RAIL to access the radio and RAIL libraries needs to be linked with Wi-SUN stack. RAIL has separate libraries for each device family and for single- and multi-protocol environments. RAIL libraries are provided in the Gecko SDK Suite. For more information refer to UG103.13: RAIL Fundamentals and other RAIL documentation.

2.2.3 EMLIB and EMDRV

The Wi-SUN stack uses EMLIB and EMDRV libraries to access EFR32 hardware. EMLIB and EMDRV peripheral libraries are provided in source code, and they must be included in the project. EMLIB and EMDRV are part of the Gecko SDK Suite. For more details on EMLIB and EMDRV, see platform EMDRV documentation and EMLIB documentation on https://docs.silabs.com.

2.2.4 Mbed TLS

The Wi-SUN stack uses the Mbed TLS library for cryptographic operations. The Mbed TLS library is provided in source code and must be included in the project. Mbed TLS is part of the Gecko SDK Suite. For more details, refer to the Mbed TLS documentation.
2.3 Optional Software Components

In addition to the Wi-SUN stack core functionality, the Wi-SUN SDK contains optional software components that you can leverage to customize the application. Add those components in the SOFTWARE COMPONENTS tab of a project, as shown in the following figure:

Figure 2.2. Wi-SUN Software Components

There are four important components:

- **Application core**: Provides application basic functionalities like event handling, callback management, and the Wi-SUN network connection.
- **CoAP**: Provides a Constrained Application Protocol (CoAP) implementation running on top of the Wi-SUN stack. The CoAP component should be used as an example of implementation of other software libraries on top of the Silicon Labs Wi-SUN stack.
- **iPerf**: Provides a widely supported tool to evaluate throughput on IP interfaces. The implementation can interoperate with other iPerf2 implementations. It currently only supports UDP server and client modes.
- **POSIX-Compliant Socket**: Provides a POSIX-like socket API on top of the standard Wi-SUN stack socket API. In addition to the API abstraction, this component makes the socket accesses thread-safe.

For the complete software documentation, visit [https://docs.silabs.com/wisun/latest/wisun-stack-api/sl-wisun-services](https://docs.silabs.com/wisun/latest/wisun-stack-api/sl-wisun-services) .
3 Wi-SUN Application Development

To get started with Wi-SUN application development, Silicon Labs recommends that you become familiar with different Wi-SUN sample applications. Then, you can use the Wi-SUN SoC Empty sample application as a template and a starting point for a new application.

The development of a Wi-SUN application consists of two main steps:
1. Responding to the events raised by the Wi-SUN stack.
2. Implementing additional application logic.

Optionally, you can change several Wi-SUN application settings with a few clicks:
1. Operating system used by the application.
2. IDE (Integrated Development Environment) used during the development.

3.1 Responding to Wi-SUN Events

A Wi-SUN application is event-driven. The Wi-SUN stack generates events when a connection is successful, data has been sent, or an IP packet is received. The application has to handle these events in the `sl_wisun_on_event()` function. The prototype of this function is implemented in `app.c`. To handle more events, the switch-case statement of this function can be created and extended. For the list of Wi-SUN events, visit [https://docs.silabs.com](https://docs.silabs.com).

3.2 Implementing Application Logic

Additional application logic can be implemented in the `app_task()` function, defined in `app.c`. The `app_task()` function is called once after the device is booted and the Wi-SUN stack is initialized. Most Wi-SUN applications’ first step is to call `sl_wisun_connect()` to connect the Wi-SUN device to a Wi-SUN border router. The remaining implementation is up to the developer. Visit [https://docs.silabs.com](https://docs.silabs.com) to check the list of Wi-SUN APIs available to the application.

3.3 Changing Operating System

Simplicity Studio 5 provides the ability to easily replace software components. This feature is leveraged to change the Real-Time Operating System (RTOS) used by the application and the Wi-SUN stack. To change the RTOS, complete these steps:
1. Go to the project SOFTWARE COMPONENTS tab.
2. Uninstall the Micrium OS Kernel component (default RTOS).
3. Install the **FreeRTOS** component.

3.4 Using a Different Development Environment

In addition to Simplicity Studio 5, you can use alternative Integrated Development Environment (IDEs). To generate a GCC makefile or an IAR Embedded Workbench project:

1. Go to the **OVERVIEW** tab.
2. Scroll down to the end of the Target and Tool Settings card and click **Change Target/SDK/Generators**.
3. In the CHANGE PROJECT GENERATORS list, select the type of projects to be generated.
4. Click **Save** and wait for Simplicity Studio to generate the project.
3.5 Wi-SUN Stack Heap Requirement

The Wi-SUN stack relies on dynamic memory allocation to function. It stores received and outgoing packets, security tokens, routing information, and more. The pick memory requirement is reached during the connection to a Wi-SUN network, especially during the authentication step. The heap size is defined under the `config/sl_memory_config.h` file in the Wi-SUN sample applications. By default, the heap size is configured through the `SL_HEAP_SIZE` define and is equal to 0x10000.

```
// <o SL_HEAP_SIZE> Minimum heap size for the application.
// <i> Default: 2048
// <i> Note that this value will configure the c heap which is normally used by
// <i> malloc() and free() from the c library. The value defines a minimum heap
// <i> size that is guaranteed to be available. The available heap may be larger
// <i> to make use of any memory that would otherwise remain unused.
#ifndef SL_HEAP_SIZE
#define SL_HEAP_SIZE   0x10000
#endif
```

This heap size is largely inflated to accommodate potential application level requirements. The bare minimum heap size recommended to run the Wi-SUN stack is 0xC000.

In addition to the standard heap size requirement, the Wi-SUN stack relies on an RTOS: Micrium OS or FreeRTOS. The stack requires a number of tasks, queues, mutexes to be created. The size of this memory pool is defined either by:

- `LIB_MEM_CFG_HEAP_SIZE` when using Micrium OS
- `configTOTAL_HEAP_SIZE` when using FreeRTOS when using the heap_4 implementation
4 Wi-SUN Configurator

When creating a new Wi-SUN sample application, a Wi-SUN Configurator is added to the project by default. It provides a configuration to the main settings of the Wi-SUN application through three panels: Application, Security, and Radio. The Wi-SUN Configurator tab is available when a project is created or can be displayed by opening `/config/wisun/wisun_settings.wisunconf`.

4.1 Application Panel

The Application panel exposes multiple Wi-SUN stack settings associated with the application. It allows editing the following, among other things:

- The network name the device will try to connect to
- The network size setting
- The device’s TX output power
- The unicast dwell interval
4.2 Security Panel

The Security panel displays the private key and certificates used by the device to authenticate itself when connecting to a Wi-SUN network. By default, it uses the Silicon Labs demonstration samples. They can be modified to use a distinct certificate infrastructure aligned with the border router certificate.

4.3 Radio Panel

The Radio panel is an interface to configure the radio profiles included in your application. It provides a user interface to access any specified Wi-SUN FAN 1.0 PHY. The complete list can be filtered to help you find the right PHY configuration. An application can embed several PHYs from different regions. The PHY used by the stack is defined by the `sl_wisun_connect()` API call.
For more information about the Wi-SUN Configurator, click **View Manual**.

Depending on the frequency band supported by the EFR32 radio board used, a default radio configuration is selected. For example, a BRD4163A radio board supporting the 868 MHz band defaults to the mandatory Wi-SUN PHY for the European region (that is, Wi-SUN FAN EU - 1 – 1a). On the other hand, a BRD4164A radio board supporting the 915 MHz band defaults to the North America mandatory Wi-SUN PHY (that is, Wi-SUN FAN NA - 1 – 1b).

The selected Wi-SUN PHY for a given project (stored in the file `radio-settings.radioconf`) can be checked in the Radio Configurator by clicking the pen icon, as shown in the following figure.

![Figure 4.1. Radio Configurator](image)

A PHY can be edited in the Radio Configurator to step out of the Wi-SUN FAN configuration (change the number of channels or frequencies).

### 4.4 Changing the Default Wi-SUN Radio Configuration

If a Wi-SUN application needs to use a different Wi-SUN PHY, use the Radio Configurator to select another one. In the **General Settings** card, open the **Select radio PHY** list. Select a new Wi-SUN PHY in the list. Keep in mind the Wi-SUN PHY selected should match the radio board capabilities. Silicon Labs does not recommend using a Chinese Wi-SUN PHY (470 MHz) on a radio board supporting the 868 MHz band.

The Wi-SUN CLI project, the network performance application, the RCP sample application, and the border router demo embed all the Wi-SUN PHYs listed in the Radio panel list. This supports changing the PHY dynamically using the CLI interface. The other sample applications only embed a single PHY by default. In this case, the PHY change must be done before project compilation through the Wi-SUN Configurator.
5 Testing and Debugging

5.1 Access Debug Traces from the Wi-SUN Stack

The Wi-SUN stack provides a logging mechanism based on the Segger RTT feature to allow a finer tracing capability. To access the Wi-SUN stack RTT traces:

1. Install the J-Link RTT Viewer.
2. Open the J-Link RTT Viewer.
3. In the Configuration panel, Connection to J-Link section, select USB.
4. In the Specify Target Device list, select the connected part (for example EFR32MG12PXXXF1024).
5. In the Target Interface & Speed panel, select JTAG and 4000 kHz.
6. In the RTT Control Block panel, select Auto Detection.
7. Click OK.
8. If you have several boards connected, a list appears.
9. Select a WSTK board running the Wi-SUN stack (border router or node).
10. Click OK.

A terminal opens and the Wi-SUN stack traces are output as shown below.

```
[DBG][wisun]: net_init_core: 0
[DBG][wisun]: sli_wisun_task_event_handler_id: 2
[DBG][SLRF]: sli_wisun_driver_register()
[DBG][SLRF]: sli_wisun_driver_register() - driver_id: 0
[DBG][SLRF]: rf_address_write: PHY_MAC_64BIT: 00:0d:6f:ff:fe:20:bd:95
[DBG][mme]: SW-MAC driver support rf extension 50000 symbol/seconds  20 us symbol time length
[DBG][swm ]: Set MAC mode to IEEE 802.15.4-2011, MTU size: 127
[DBG][SLRF]: rf_address_write: PHY_MAC_64BIT: 00:0d:6f:ff:fe:20:bd:95
[DBG][wisun]: arm_nwk_interface_lowpan_init: 1
```

These logs can be used to report an issue to Silicon Labs support.

5.2 Export Wi-SUN Traces to Wireshark

The Wi-SUN traces export feature requires Simplicity Studio 5.1.0 or higher.

Simplicity Studio’s Network Analyzer enables debugging of complex wireless systems on a number of Silicon Labs part families. Network Analyzer includes a partial Wi-SUN protocol analyzer (that is, the Wi-SUN payload cannot be decrypted). However, it can be used to export traces to another analyzer like Wireshark.

To export Wi-SUN traces with the Network Analyzer to Wireshark, install Wireshark and follow this procedure in Simplicity Studio 5:

1. In the Simplicity IDE perspective, Debug Adapter view, right-click an EFR32xG12 running the Wi-SUN stack.
2. Select Start Capture.
3. A Live tab opens in the Editor area. It traces packets sent and received by the Wi-SUN device.
4. When you have traced the communication, click Export.

5. Under Select export format, select PCAP NG exporter.
6. Enter a path and a file name in which to store the trace.
7. In **Export mode**, select **Wi-SUN (auto-detect protocol)**.
8. Click **OK**.

Open the new file in Wireshark. Wireshark should automatically analyze the file as a Wi-SUN exchange. The communication is initially encrypted thanks to the Wi-SUN encryption protocol. To decrypt the communications, the GAK key and key index set information are required. They can be retrieved on the border router CLI by issuing the following command:

```bash
> wisun get wisun

wisun.state = started (2)
wisun.network_name = "juju"
wisun.regulatory_domain = EU (3)
wisun.operating_class = 1
wisun.operating_mode = 0x1a
wisun.network_size = test (4)
wisun.tx_power = 20
wisun.gak2 = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
wisun.gak3 = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
wisun.gak4 = 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
wisun.ip_addresses = [ll: fe80::20d:6fff:fe20:bd95 gua: fd00:7283:7e00:0:20d:6fff:fe20:bd95
dodagid: fd00:6172:6d00:0:20d:6fff:fe20:bd95]

Silicon Labs Wi-SUN devices in the network use `wisun.gak1` as the GAK key. In the capture above, the GAK key is `d6:5c:fb:3a:8a:98:78:32:85:9b:ff:1a:b8:dc:5d:de` and the **key index** is **1**.

In Wireshark:
1. Click **Edit**.
2. Click **Preferences**...
3. Expand the **Protocols** list and select **IEEE 802.15.4**.
4. Next to Decryption Keys, click **Edit**.
5. In the Keys window, click + (plus).
6. Under **Decryption key** enter the GAK key, and under **Decryption key index** enter the key index.
7. Click **OK**.
Wireshark is now able to decrypt the traces and the higher-level protocols (ICMP, TCP, UDP...). The following example of traces shows a router pinging its border router.

<table>
<thead>
<tr>
<th>Time</th>
<th>Source MAC</th>
<th>Destination MAC</th>
<th>Payload Length</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>348</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>0.0255460000</td>
<td>Wi-SUN 105</td>
</tr>
<tr>
<td>349</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>0.0256050000</td>
<td>Wi-SUN 105</td>
</tr>
<tr>
<td>350</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>00:0d:6f:ff:fe:20:b6:f9</td>
<td>0.0255490000</td>
<td>Wi-SUN 105</td>
</tr>
<tr>
<td>351</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>25.953172000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>352</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0290880000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>353</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.4209990000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>354</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0209000000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>355</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.2015710000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>356</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0290880000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>357</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>1.4897450000</td>
<td>Wi-SUN 105</td>
</tr>
<tr>
<td>358</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>1.4809420000</td>
<td>Wi-SUN 105</td>
</tr>
<tr>
<td>359</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>3.8548340000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>360</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0291430000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>361</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0177590000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>362</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0266660000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>363</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>3.2415580000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>364</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0553600000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>365</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.4192200000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>366</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0266660000</td>
<td>Wi-SUN 44</td>
</tr>
<tr>
<td>367</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.3066700000</td>
<td>ICMPv6 151</td>
</tr>
<tr>
<td>368</td>
<td>f0:00:72:83:7e:00:0:20:6ff:fe_f0:00:61:72:6d:0:0:20:6ff:fe20:bd:45</td>
<td>00:00:6f:ff:fe:20:b6:f9</td>
<td>0.0691370000</td>
<td>Wi-SUN 44</td>
</tr>
</tbody>
</table>

The PTI output is limited in number of bytes per messages. Packets above 1022 bytes are truncated using the WSTK Firmware 1v4p0 or later (200 bytes with earlier versions).

5.3 Connect the Wi-SUN Network to Another IP Network

Refer to the steps in AN1332: Silicon Labs Wi-SUN Network Setup and Configuration to open a backhaul connection from the Wi-SUN border router.
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

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and “Typical” parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice to the product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Without prior notification, Silicon Labs may update product firmware during the manufacturing process for security or reliability reasons. Such changes will not alter the specifications or the performance of the product. Silicon Labs shall have no liability for the consequences of use of the information supplied in this document. This document does not imply or expressly grant any license to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any FDA Class III devices, applications for which FDA premarket approval is required or Life Support Systems without the specific written consent of Silicon Labs. Silicon Labs disclaims all express and implied warranties and shall not be responsible or liable for any injuries or damages related to use of a Silicon Labs product in such unauthorized applications. Note: This content may contain offensive terminology that is now obsolete. Silicon Labs is replacing these terms with inclusive language wherever possible. For more information, visit www.silabs.com/about-us/inclusive-lexicon-project

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, “the world’s most energy friendly microcontrollers”, Redpine Signals®, WiSeConnect®, n-Link, ThreadArch®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, Gecko OS, Gecko OS Studio, Precision32®, Simplicity Studio®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri, the Zentri logo and Zentri DMS, Z-Wave®, and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. Wi-Fi is a registered trademark of the Wi-Fi Alliance. All other products or brand names mentioned herein are trademarks of their respective holders.