Silicon Labs Thread 2.2.1

Silicon Laboratories, Inc.

April 7, 2017

Note **SDK_HOME** refers to the location where the Silicon Labs SDK is installed on your machine.

THREAD_HOME refers to the location where the Silicon Labs Thread stack is installed on your machine. For example, **SDK_HOME**/protocol/thread_2.2.

1 Introduction

Thread is a secure, wireless mesh networking protocol. It is an open standard built upon a collection of existing IEEE and IETF standards, developed by Silicon Labs in conjunction with the Thread Group (http://www.threadgroup.org/).

Thread supports IPv6 addresses and provides low-cost routing to other IP networks. It is optimized for low-power and battery-packed operations, and also wireless device-to-device communication. It is specifically designed for Connected Home applications where IP-based networking is desired.

Please see **THREAD_HOME**/documentation/UG103-11-AppDevFundamentals-Thread.pdf for detailed information about the stack.

If you are a first-time user, see QSG113, Getting Started with Silicon Labs Thread, for instructions on installing and configuring your development environment, building and flashing a sample application, and documentation references pointing to next steps.

For information on the Silicon Labs Thread Border Router Kit, how to set

it up with an existing EM35x Development Kit, and how to demonstrate an example application, see QSG102, Thread Border Router Add-On Kit Quick Start Guide.

For information on how to build your own border router applications, see UG116, Developing Custom Border Router Applications.

2 GA release

2.1 What's New

- Improved CoAP interface
- Improvements and bug fixes for Thread 1.1 interoperability
- Support for EFR32MG12
- Support for Gecko Bootloader

2.2 Software Components

The stack software release is intended to be used with the following versions of other components:

- Simplicity Studio version 4.11 or later: Downloaded from http://www.silabs.com/simplicity.
  NOTE: If you have an earlier version of Simplicity Studio installed, remove it before installing the newer Simplicity Studio version.
- IAR Embedded Workbench for ARM 7.80.2, used as a compiler in the Simplicity Studio development environment.
  If you have an EFR32MG WSTK (Wireless Starter Kit), you can download a 30-day evaluation version of EWARM from the Silicon Labs Technical Support portal http://www.silabs.com/support. If you have an EM35x Development Kit, then an IAR Embedded Workbench Download Card is included with that kit. For EM35X-DEV, you will receive a 30-day evaluation version of EWARM. For EM35X-DEV-IAR,

you will receive a licensed version and a Welcome Letter with full development license information. To install the IAR Embedded Workbench for ARM:

  - Download the installer from the link provided in the license information. The executable will extract files to a temporary folder and then launch the installer. Click on the "Install IAR EWARM Embedded Workbench" link and follow the install instructions.
  - Refer to the "Installation and licensing information" section of the IAR installer for additional information about the installation process and how to configure your license.

  If you are receiving this release as a product upgrade, the installer for IAR EWARM 7.80.2 is available at ftp://files.iar.com/pub/silabs/EWARM-7802.exe.

- Utilities
  - ISA3 Utilities version 4.8.0.0: Downloaded from the Silicon Labs support portal. Installs the ISA3 drivers so that the debug adapter (ISA3) can be used both over USB and Ethernet, along with other development and production utilities. Always run the ISA3 Utilities installer after all previous installers, including after IAR-EWARM.
    The installer modifies your PATH environment variable so that the command line utilities can be easily executed from a Windows Command Prompt. See UG107, EM35x Utilities Guide: For the EM35x SoC Platform, for detailed information on the EM35x utilities.
    NOTE: ISA3 Utilities are also required for the Mighty Gecko WSTK.
  - For the Mighty Gecko WSTK: Simplicity Commander, installed along with Simplicity Studio in a subfolder of the install folder. See UG162, Simplicity Commander Reference Guide, for more information.

2.3 Package contents

The Silicon Labs Thread release package contains the following directories under **SDK_HOME**:

- **app/**
  - **esf_common/**
    Contains template and metadata files used by the Application Framework

- **hardware/**
  Contains source describing how development kit and reference design boards work
  - **kits/**
  - **reference_design/**
- **platform/**
  - **base/**
    - **hal/**
      Contains board headers and other include headers pertaining to the HAL library.
  - **CMSIS/**
  - **Device/**
  - **emdrv/**
  - **emlib/**
  - **middleware/**
    - **glib/**
  - **radio/**
    - **rail_lib/**
- **protocol/**
  - **thread_2.2/**
    - **app/**
      Contains application framework files and application utilities (such as serial and command interpreter utilities) that aid in developing a sample application that uses the Silicon Labs Thread stack.
    - **build/**
      Contains the following libraries:
      - HAL library

        - Manufacturing library
        - NCP library
        - Simulated EEPROM library
        - Silicon Labs Thread stack library, containing stack functionality
        - mbedTLS library
      * **documentation/**
        Data sheets, application development fundamentals, and other important reading material.
      * **ncp-images/**
        Contains binary images for use on a network coprocessor (NCP).
      * **stack/**
        Contains stack include headers that are necessary for building a sample application.
      * **tool/**
        Contains bootloaders.
- **util/**
  - **third_party/**
    * **mbedtls/**

2.4 Installation

First-time users, see QSG113, Getting Started with Silicon Labs Thread, for step-by-step instructions on installing software, configuring the Simplicity Studio development environment, and building and flashing an example application.

- EM35x Development Kit: Refer to the Quick Start Guide included in your Development Kit for detailed step-by-step instructions about initially setting up your kit.

- EFR32MG Mesh Networking Kit: see QSG113, Getting Started with Silicon Labs Thread, for instructions.

2.5 Stack API documentation

The Silicon Labs Thread stack API reference guide is Doxygen-generated documentation that is rooted at **THREAD_HOME**/documentation/Thread-Doxygen/index.html. This document contains a comprehensive list of APIs used to interface to the Thread network. These APIs concern network management, device and stack management (including management of stack tables, event scheduling, message buffers and security), messaging, fragmentation, serial communication, token access, peripheral access, bootload utilities, etc.

3 Known issues

3.1 General

- At the point of this release, the Thread specification and certification process is not 100 percent complete so the stack and related APIs may change without notice.
- For this release, we are including a preliminary application layer that will be fleshed out further over time. Please note that this application layer will change significantly before final release.
- Attempting to bootload an EM35xx over SPI is known to have issues.
- The Client-Sleepy sample application, when running on an EFR32, has a baseline of 10ua current. This is approximately 7ua higher than what is possible due to the external flash being powered on. This will be fixed in a future release. If you need help reducing this, please contact support.

4 Deprecated APIs and Functionality

- Thread and Zigbee are migrating to an over-the-air implementation for the application layer that will leverage CoAPs Blockwise transfer feature. As such, we plan to deprecate our TFTP example starting with our Q2 SDK release since that will be non-standard.

5 Fixed issues in 2.2.1

- Fixed an issue where the first byte of the mfglib test packet was being dropped when the packet was passed from the NCP to the host.
- The border router now clears the global IP addresses on the host after a network reset on NCP.

6 Fixed issues in 2.2.0

- The CoAP API has been significantly improved. Please refer to the notes within the "Detailed Description" section of the Coap module in the doxygen-generated documentation which is rooted at **THREAD_HOME**/documentation Doxygen/index.html. These notes provide a brief overview of the key parts of the API as well as tips on migrating from the old API to the new API. The API is also fully documented in the C header file, **THREAD_HOME**/stack/include/coap.h. Here is a list of some of the improvements.
  - Internal CoAP functions have been removed from stack/include/coap.h.
  - CoAP responses can now be sent outside of the incoming message handler.
  - Applications will now receive all responses to requests sent to a multicast address.
  - Separate responses, as described in section 5.2.2 of RFC-7252, can now be sent using the improved CoAP API.

- The client and client-sleepy sample applications have been modified such that joining will only occur after a button presss or after invoking the new join CLI command. In addition, these applications now generate a random join key and store it in the new THREAD_JOIN_KEY manufacturing token.
- The host ip-driver-app has been modified to work on Linux Kernel 4.4.16.
- A new application framework plugin, thread-test-harness-cli, has been added. This plugin provides all of the CLI required for testing against the GRL Thread Test Harness. This plugin is currently in beta, and is not guaranteed to pass all tests when run against the GRL Thread Test Harness.
- Compression of the link local multicast address has been corrected.
- In addition the the CoAP API changes described previously the following APIs have also changed.
  - New counters introduced.
    * EMBER_COUNTER_PTA_LO_PRI_REQUESTED - The number of times a low priority packet traffic arbitration request has been made.
    * EMBER_COUNTER_PTA_HI_PRI_REQUESTED - The number of times a high priority packet traffic arbitration request has been made.
    * EMBER_COUNTER_PTA_LO_PRI_DENIED - The number of times a low priority packet traffic arbitration request has been denied.
    * EMBER_COUNTER_PTA_HI_PRI_DENIED - The number of times a high priority packet traffic arbitration request has been denied.
    * EMBER_COUNTER_PTA_LO_PRI_TX_ABORTED - The number of times a low priority packet traffic arbitration transmission has been aborted.

    - EMBER_COUNTER_PTA_HI_PRI_TX_ABORTED - The number of times a high priority packet traffic arbitration transmission has been aborted.
  - New APIs for handling network data. These are described in more detail in the C header file located at **THREAD_HOME**/stack/include/network-management.h.
    - emberGetNetworkData
    - emberGetNetworkDataReturn
    - emberNetworkDataChangeHandler

7 Fixed issues in 2.1.1

- The default child timeout for sleepy end devices was changed from 300 seconds to 240 seconds. This aligns with the Thread 1.1 specification.
- Added new API emberIsIpv6UnspecifiedAddress() which checks if a given EmberIpv6Address is set to all zeroes which represents an unspecified address.
- When emberGetRipEntry() is called with an 0xFF index to request all valid RIP table entries the stack will return valid entries through calls to emberGetRipEntryReturn() as it always has but now once all valid entries have been returned an extra zeroed-out entry is returned to indicate completion.
- When emberGetGlobalPrefixes() is called to request the list of global prefixes that we know about the stack will return valid entries through calls to emberGetGlobalPrefixReturn() as it always has but now once all valid entries have been returned an extra zeroed-out entry is returned to indicate completion.
- When emberGetDhcpClients() is called to request the list of DHCP clients that we know about the stack will return valid entries through calls to emberGetDhcpClientReturn() as it always has but now once all valid entries have been returned an extra zeroed-out entry is returned to indicate completion.

- When emberGetGlobalAddresses() is called to request list of global addresses configured on this device the stack will return valid entries through calls to emberGetGlobalAddressReturn() as it always has but now once all valid entries have been returned an extra zeroed-out entry (an IPv6 unpspecified address) is returned to indicate completion.
- Fixed an issue introduced in Silicon Labs Thread 2.1.0 where the stack was dropping data requests with security enabled and thus preventing some sleepy end devices from sleeping.

8 Fixed issues in 2.1.0

- Added new API emberGetPtaOptions() and corresponding callback emberGetPtaOptionsReturn() to get packet traffic arbitration configuration options.
- Added new API emberSetPtaOptions() and corresponding callback emberSetPtaOptionsReturn() to configure packet traffic arbitration options.
- The following API functions have been renamed:
  - emberSendJoinerEntrust to emberSendEntrust
  - emberSetBeaconSteeringData to emberSetSteeringData

9 Fixed issues in 2.0.0

- DHCP is now an optional component within the Silicon Labs Thread stack. To include support for DHCP applications must link against dhcp-library.a. To exclude support for DHCP and thus save flash space applications should link against dhcp-stub-library.a.
- SW flow control is now supported between a host and NCP when connected through a UART interface. To enable, define EMBER_APPLICATION_USES_SOFTW and EMBER_SERIAL1_XONXOFF.

- Added new API emberSetRadioHoldOff() and corresponding callback emberSetRadioHoldOffReturn() which gives both host and SoC applications the ability to enable or disable radio holdoff. When radio holdoff is enabled it configures RHO_GPIO in BOARD_HEADER as an input which, when asserted, will prevent the radio from transmitting. When radio holdoff is disabled it configures RHO_GPIO for its original default purpose.
- Added new API emberGetPtaEnable() and corresponding callback emberGetPtaEnableReturn() to get whether packet traffic arbitration is enabled or disabled.
- Added new API emberSetPtaEnable() and corresponding callback emberSetPtaEnableReturn() to enable or disable packet traffic arbitration.
- Added new API emberGetAntennaMode() and corresponding callback emberGetAntennaModeReturn() to get the current antenna mode.
- Added new API emberSetAntennaMode() and corresponding callback emberSetAntennaModeReturn() to set the current antenna mode.
- Added new callback emberCounterValueHandler() which is invoked to query the application for the counter value of an event defined by the given EmberCounterType.
- An additional parameter, isStable, has been added to the emberGetGlobalPrefixReturn() callback.
- Added support for EMBER_MINIMAL_END_DEVICE, an always-on end device like EMBER_END_DEVICE, but IP address discovery is performed by the parent on its behalf to help it conserve resources.
- Added the following new APIs to convert IPv6 addresses and prefixes to and from strings.
  - emberIpv6AddressToString()
  - emberIpv6PrefixToString()
  - emberIpv6StringToAddress()
  - emberIpv6StringToPrefix()

- Added new API emberRadioGetRandomNumbers() and corresponding callback emberRadioGetRandomNumbersReturn() to obtain true random numbers.
- The following callback functions have been renamed:
  - emberDhcpServerChange to emberDhcpServerChangeHandler
  - emberAddressConfigurationChange to emberAddressConfigurationChangeHanlder
  - emberExternalRouteChange to emberExternalRouteChangeHandler
  - emberMarkApplicationBuffers to emberMarkApplicationBuffersHandler
  - emberSlaacServerChange to emberSlaacServerChangeHandler
  - emberDeepSleepCallback to emberDeepSleepCompleteHandler
- Address prefix length is now in bits instead of bytes. The following APIs and callbacks are affected by this change:
  - emberConfigureGateway()
  - emberConfigureExternalRoute()
  - emberGetGlobalPrefixReturn()
  - emberDhcpServerChangeHandler()
  - emberRequestDhcpAddress()
  - emberRequestDhcpAddressReturn()
  - emberSlaacServerChangeHandler()
  - emberRequestSlaacAddress()
  - emberRequestSlaacAddressReturn()
  - emberGetGlobalAddresses()
  - emberExternalRouteChangeHandler()
- The syntax for specifying prefixes in CLI commands has changed. The format for a prefix is now the first address in the block, a slash, and

a decimal value equal to the size of the prefix in bits (e.g., fe80::/64, 2001:db8:1234::/48). The following CLI commands are affected by this change:

  - network-management commission ...
  - network-management form ...
  - network-management gateway ...
  - network-management global-addresses ...

10 Fixed issues in 1.0.7

- Added a new API emberEnableHostJoinClient() so applications can run the commissioning state machine on the host instead of the NCP. The behavior of this API is documented in stack/include/network-management.h.

11 Fixed issues in 1.0.6

- Added a new API emberSetCtune() and corresponding callback emberSetCtuneReturn() to change the CTUNE value. Involves switching to HFRCO and turning off the HFXO temporarily. (Only valid on EFR32.)
- Added a new API emberGetCtune() and corresponding callback emberGetCtuneReturn() to get the CTUNE value. (Only valid on EFR32.)
- Added support to build NCP applications with optional extensions to initialization, main loop processing, event definition and handling, and host/NCP commands.
- Added NCP SPI and NCP UART sample applications. It is recommended these be used as the starting point for building customized NCP applications.

12 Fixed issues in 1.0.5

- Join failure argument added to emberNetworkStatusHandler() callback.
- Host callback function emberStateReturn will now always be called after a call to emberState. Host applications must now implement emberStateReturn().
- Messages sent to the "All Mesh Nodes" multicast address (ff03::1) are no longer delivered to rx-off-when-idle Children. The "All Thread Nodes" address (ff33:40:¡ula prefix¿:1) can be used to multicast to these devices.

13 Fixed issues in 1.0.4

- Added EFR32 SoC and NCP support.
- Added border router sample application.
- Added support for NCPs over SPI.
- Added standalone bootloaders to the SPI and UART NCPs. The included NCP images now must be used with ezsp-spi-bootloader for SPI or serial-uart-bootloader for UART.
- Added a new API emberGetStandaloneBootloaderInfo() and corresponding callback emberGetStandaloneBootloaderInfoReturn() to obtain the version of the installed standalone bootloader as well as the platform, micro, and PHY of the NCP.
- Added a new API emberLaunchStandaloneBootloader() and corresponding callback emberLaunchStandaloneBootloaderReturn() to inform the NCP to launch the standalone bootloader.
- Added a new API emberGetMfgToken() and corresponding callback emberGetMfgTokenReturn() to obtain a manufacturer token.
- Added a new API emberSetMfgToken() and corresponding callback emberSetMfgTokenReturn() to set a manufacturer token.

- The way the network status is communicated to the application has been made more consistent. There is a new callback, emberNetworkStatusHandler(), that is called whenever the network status changes. The return callbacks for the form, join, resume, and attach commands now indicated only whether the process was initiated. There is a new network status type, EMBER_JOINED_NETWORK_ATTACHING, for use when the node is attaching. The name of EMBER_JOINED_NETWORK has also been changed to EMBER_JOINED_NETWORK_ATTACHED. The new callback behavior is documented in stack/include/network-management.h.
- The mbedTLS functionality required by the Silicon Labs Thread stack is now released as a separate library, mbedtls-library.a. This new library must be included when linking an application that includes the Silicon Labs Thread stack.
- Changed emberCoapMessageHandler to have a single parameter, a pointer to an EmberCoapMessage struct, which replaces the previous discrete parameters. EmberCoapMessage also replaced EmberAfCoapDispatchRequest in the CoAP Dispatch plugin.
- Added a macExtendedId parameter to emberStateReturn.

14 Fixed issues in 1.0.3

- Changed some constant names in stack/include/network-management.h to reflect their actual purpose.
  EmberJoiningMode:
  - **EMBER_JOINING_NO_STEERING** is now **EMBER_JOINING_ALLOW_ALL_STEERING**
  - **EMBER_JOINING_CLEAR_STEERING** is now **EMBER_JOINING_ALLOW_EUI_STEERING**
  - **EMBER_JOINING_CLEAR_STEERING_SMALL_EUI64** is now **EMBER_JOINING_ALLOW_SMALL_EUI_STEERING**

Flag value for emberCommissionerStatusHandler:

  - **EMBER_JOINING_WITH_STEERING** is now **EMBER_JOINING_WITH_EUI_STEERING**

- Added a new API emberConfigureExternalRoute() to define an external route set, which is a route for a Thread network IPv6 packet that must traverse a border router and be forwarded to an exterior network.
- Removed a macExtendedId field mistakenly added to the Silicon Labs Thread token struct in the 1.0.2 release. NOTE: this breaks token compatibility between 1.0.2 and 1.0.3, so existing networks cannot be upgraded in place and must be factory reset.
- Removed options and hop limit arguments to emberSendUdp() and its associated CLI commands.
- Various minor cleanup tasks.

15 Fixed issues in 1.0.2

- Added a destination IP address as the second argument to emberCoapMessageHandler().
- Fixed a bug in which a sleepy end device would change to a powered end device if it failed to join.
- Added version and steering fields to the EmberMacBeaconData struct.
- Fixed a problem in which GUA addresses were being lost on reset.
- When configuring a border router as a DHCP server:
  The preferred lifetime should lie between **EMBER_MIN_PREFERRED_LIFETIME_SEC** and **EMBER_MAX_LIFETIME_DELAY_SEC**.
  The valid lifetime should lie between **EMBER_MIN_VALID_LIFETIME_SEC** and **EMBER_MAX_LIFETIME_DELAY_SEC**.
  As always, preferred and valid lifetimes are ignored for SLAAC prefixes.
- Fixed multicast relay timing to conform to the Thread specification.

- Various other minor bug fixes found during testing.

16 Fixed issues in 1.0.1.1

- An include file problem, which prevented host applications from compiling, has been resolved.
- A bug has been fixed that caused multicast messages to silently fail when attempting to send from a host to the mesh.